POLITECNICO DI MILANO

Scuola di Ingegneria dell'Informazione

Dipartimento di Elettronica e Informazione

Corso di Laurea Specialistica in Ingegneria Informatica

VALUTAZIONE SPERIMENTALE DI UN APPROCCIO AD AGENTI COOPERATIVI PER LA GESTIONE DI RETI ELETTRICHE

Relatore: prof. Francesco AMIGONI

Correlatori: ing. Michele TARTARA

ing. Vito Giovanni CASTELLANA

Tesi di Laurea Specialistica di:

Alessandro BELLEMO

Matr. n. 708673

Anno Accademico 2010-2011

Sommario

La gestione efficiente delle reti elettriche sta assumendo un'importanza sempre maggiore a causa dell'aumento del fabbisogno energetico mondiale e delle problematiche ambientali ad esso associate; per questo motivo sono stati creati vari modelli per la gestione automatica delle reti elettriche, alcuni dei quali basati su sistemi multiagente. I modelli proposti, però, sono spesso troppo specifici e si applicano solo ad alcuni tipi di reti elettriche, oppure hanno dei comportamenti non desiderati quando è necessario reagire a situazioni di guasto o a periodi di picco.

Lo scopo della tesi è valutare un nuovo modello per la gestione delle reti elettriche basato sul formalismo DCOP, Distributed Constraint Optimization Problem.

Grazie alla suite di programmi FRODO sono stati risolti i DCOP che modellano alcune reti elettriche e sono state valutate le prestazioni ottenute al variare degli algoritmi risolutivi, quali ad esempio il tempo di esecuzione e la quantità di messaggi scambiati dagli agenti; sono state inoltre individuate alcune criticità nella soluzione dei problemi e sono state fornite delle spiegazioni a questi comportamenti indesiderati degli algoritmi.

Ringraziamenti

Un sentito grazie, innanzitutto, ai miei genitori e a mia sorella Elisa, che mi hanno sopportato così a lungo, nonostante il mio carattere a volte un po' bizzoso.
Non può mancare nei ringraziamenti mia nonna Cesira, che festeggia ogni mia visita preparando succulenti manicaretti di cui mai mi stancherò.
Un enorme grazie a mia nonna Tilde, che per tanti anni si è amorevolmente presa cura di me.
Grazie a mio zio Alessandro, che è sempre pronto a fornirmi supporto medico che io tento di ricambiare con supporto informatico.
Grazie allo zio Eros e alla zia Isabella.
Grazie, infine, a Marcel, compagno di tante bevute e di qualche giro di pista a Monza. Possa tu vendere auto da 100.000€ ogni giorno.

Indice

Sommario....3
Ringraziamenti....5
Indice....7
Capitolo 1....9
Capitolo 2....11
2.1 Il problema della gestione delle reti elettriche....11
2.2 Stato dell'arte....12
2.3 Modello proposto....13
2.4 Implementazione del modello....15
2.5 Applicazioni del modello....16
Capitolo 3....19
3.1 Il DCOP usato dal modello....19
3.2 Classificazione dei vincoli....20
3.3 Funzioni di costo....22
Capitolo 4....25
4.1 Informazioni generali su FRODO....25
4.2 Modalità di funzionamento....26
4.3 Parser utilizzabili da FRODO....26
4.4 Metriche di valutazione degli algoritmi....27
4.5 Classificazione degli algoritmi offerti da FRODO....28
4.6 Limitazioni di FRODO e loro impatto sul modello....29
4.7 Formato dei file contenenti DCOP risolvibili da FRODO....30
Capitolo 5....37
5.1 Descrizione degli esperimenti eseguiti....37
5.2 Analisi delle prestazioni in base al numero di agenti....40
5.3 Analisi delle prestazioni in base alla cardinalità del dominio....46
5.4 Analisi delle prestazioni in base alle classi di priorità....49
5.5 Analisi delle prestazioni in base alla potenza disponibile....51
5.6 Analisi dei messaggi scambiati in base alla cardinalità del dominio....53
5.7 Analisi dei messaggi scambiati in base alle classi di priorità....55
5.8 Analisi dei messaggi scambiati in base alla potenza disponibile....57
5.9 Analisi delle prestazioni dell'algoritmo MGM....61
Capitolo 6....65
6.1 Considerazioni sui risultati ottenuti....65
6.2 Lavori futuri....66
Bibliografia....69

Capitolo 1

La gestione efficiente delle reti elettriche sta assumendo un'importanza sempre maggiore a causa dell'aumento del fabbisogno energetico mondiale e delle problematiche ambientali ad esso associate. Si tratta di un problema di difficile soluzione perché le reti elettriche non possono essere considerate statiche, ma variano nel tempo a seconda dei carichi attivi, della potenza resa disponibile dai generatori e da eventuali guasti o malfunzionamenti. Inoltre ogni sistema per la gestione automatica delle reti elettriche deve fornire una strategia da usare quando, a causa di guasti o di picchi di richieste, non c'è potenza sufficiente a soddisfare le richieste; generalmente la strategia più diffusa consiste semplicemente nella disattivazione di carichi fino a riportare la rete a livelli stabili.
Scopo di questa tesi è valutare un modello multiagente per gestire le reti elettriche; tale modello astrae completamente dalla rete gestita ed è quindi potenzialmente adattabile ad ogni situazione.

Il modello valutato si basa sul formalismo DCOP (Distributed Constraint Optimization Problem); tali problemi consistono nella minimizzazione di una funzione di costo, in presenza di vincoli, attraverso un sistema multiagente distribuito. Nel modello, gli agenti corrispondono ai carichi della rete, le variabili controllate dagli agenti sono le potenze utilizzate dai carichi stessi e ogni variabile ha un proprio dominio finito; i vincoli, infine, servono a organizzare gli agenti in una gerarchia e ad evitare un consumo di potenza superiore a quella disponibile.
L'obiettivo della tesi è la valutazione del modello presentato, così da poter stabilire se è effettivamente in grado di gestire una rete e quali sono le sue prestazioni.
Per poter valutare il modello si è fatto uso di FRODO, una suite di programmi open source il cui scopo è risolvere DCOP. FRODO implementa vari algoritmi per la soluzione di DCOP ed è in grado di fornire non solo la soluzione al problema ma anche alcune metriche per la valutazione delle prestazioni ottenute dagli algoritmi.
Il modello è stato applicato a più configurazioni di ipotetiche reti elettriche. È stato quindi risolto utilizzando diversi algoritmi e sono stati valutati il tempo utilizzato dagli algoritmi per trovare la soluzione al problema, il numero e la dimensione dei messaggi scambiati dagli agenti e la capacità degli algoritmi di trovare la soluzione ottima al problema.

La tesi è strutturata nel modo seguente.

Nel Capitolo 2 si illustra lo stato dell'arte e si spiegano i motivi che hanno portato alla creazione di un nuovo modello per la gestione di carichi elettrici.

Il Capitolo 3 illustra le caratteristiche del modello in questione e mostra come esso sia adattabile a vari tipi di reti e in grado di gestire reti che variano nel tempo.

Il Capitolo 4 illustra brevemente FRODO, il software utilizzato per valutare il modello e le prestazioni degli algoritmi.

Il Capitolo 5 contiene una descrizione dettagliata degli esperimenti effettuati, i relativi risultati e alcune considerazioni sulle prestazioni ottenute dagli algoritmi.

Nel Capitolo 6 si riassumono le valutazioni del modello e degli algoritmi da esso utilizzati e si presentano delle considerazioni su eventuali lavori futuri.

Capitolo 2

In questo capitolo viene spiegato per quale motivo è necessario trovare un sistema per la gestione delle reti elettriche e vengono presentati alcuni accenni a modelli pre-esistenti. Successivamente viene spiegato perché è stato creato un nuovo modello e cosa lo differenzia dalle altre soluzioni.
La Sezione 2.1 introdurrà il problema della gestione delle reti elettriche.
La Sezione 2.2 fornirà alcuni cenni allo stato dell'arte e ad alcune soluzioni già pubblicate per la gestione di reti elettriche attraverso sistemi multi-agente.
La Sezione 2.3 illustra cosa differenzia il modello proposto dalle soluzioni pre-esistenti e le caratteristiche del modello valutato in questo lavoro.
La Sezione 2.4 spiega che approccio è stato scelto per implementare il modello e i vantaggi così ottenuti.
La Sezione 2.5 mostra alcuni esempi di reti che potrebbero essere gestite con il modello qui proposto e valutato.

2.1 Il problema della gestione delle reti elettriche

Negli ultimi anni il tema dell'energia elettrica e della sua gestione ha assunto un'importanza crescente ed è stato al centro di numerosi dibattiti. Il fabbisogno energetico mondiale è cresciuto rapidamente e ciò ha reso sempre più importante la generazione dell'energia e la sua distribuzione. Inoltre la maggior parte dell'energia è tuttora creata tramite l'utilizzo di fonti non rinnovabili e spesso inquinanti; ciò significa che al problema dell'approvvigionamento dell'energia si aggiunge il problema di un utilizzo consapevole e giudizioso di risorse che sono destinate a diventare sempre più rare e ad aumentare il livello di inquinamento del nostro pianeta.
Una risposta a tale problema è stato un aumento dell'utilizzo di fonti rinnovabili (sole, vento, maree, energia geotermica, ecc), che spesso hanno anche il pregio di permettere all'utente finale di creare da sé l'energia di cui necessita, senza doversi per forza rivolgere a società esterne. Alcune di queste fonti rinnovabili, però, soffrono di problemi riguardante la continuità con cui producono energia; ad esempio l'energia prodotta da generatori eolici dipende dalla velocità del vento e i pannelli solari sono totalmente inutili nelle ore notturne.

Chi utilizza queste tecnologie, quindi, può essere costretto a integrarle con i sistemi tradizionali di approvvigionamento energetico oppure a cambiare il modo con cui viene consumata l'energia nei periodi calo di produzione (ad esempio spegnendo i componenti non critici della propria rete elettrica). Un altro aspetto interessante e attualmente al centro di vari studi consiste nella possibilità di conservare l'energia in batterie per poi utilizzarla in momenti di sottoproduzione.

Il problema della gestione delle reti elettriche ha molte sfaccettature: i produttori, ad esempio, vogliono produrre una quantità di energia il più vicino possibile al fabbisogno; in questo modo possono soddisfare tute le esigenze dei consumatori finali senza investire ingenti quantità di denaro in infrastrutture che non verranno poi sfruttate.

Le società che distribuiscono l'energia hanno altri problemi: in primo luogo devono acquistare energia a sufficienza dai produttori al minor costo possibile; successivamente devono distribuire tale energia in reti spesso molto estese tenendo conto delle esigenze dei vari clienti. Gli utenti finali, infine, necessitano di un servizio che abbia alta affidabilità e costi contenuti e che sia in grado di integrarsi correttamente con eventuali generatori che utilizzano fonti rinnovabili.

2.2 Stato dell'arte

Data l'importanza di questo problema sono stati pubblicati numerosi lavori che si prefiggono lo scopo di gestire reti elettriche e la distribuzione della potenza elettrica agli utenti finali.

Ramchurn, Vytelingum, Rogers e Jennings hanno proposto un sistema multi-agente di controllo per la gestione decentralizzata demand side di Smart Grid [1]; gli autori propongono un sistema che permette agli agenti di cooperare per gestire la fornitura di corrente elettrica alle 26 milioni di case del Regno Unito in modo da evitare periodi di picco che possono causare malfunzionamenti della rete o addirittura blackout.

Penya e Jennings propongono un altro sistema multi-agente in cui gli agenti competono tra di loro per acquistare energia elettrica tramite aste organizzate dalle compagnie produttrici [2]; ogni giornata viene divisa in 24 slot da un'ora l'uno e gli agenti effettuano delle offerte per tentare di acquistare l'energia elettrica di ora in ora al prezzo più basso possibile.

Ceppi e Gatti propongono un sistema multi-agente in cui un “mercato elettronico” funge da mediatore tra le aziende che producono l'energia e i consumatori [3]; le aziende propongono il proprio prezzo di vendita al mercato, il quale acquista i lotti di energia più convenienti fino a

soddisfare la domanda di energia.
Prýmek e Horák hanno studiato un sistema multi-agente per la distribuzione di energia che privilegia l'affidabilità e che tiene conto di eventuali perdite economiche dovute a disservizi [4].
Santofimia, del Toro, Roncero-Sánchez, Moya, Martinez e Lopez hanno creato un modello multi-agente per la gestione dell'energia che pone particolare attenzione sulla qualità della corrente erogata [5]; l'approccio scelto è quello del modello comportamentale qualitativo, che permette di valutare negativamente eventuali disservizi legati alla qualità dell'energia (deformazioni della forma d'onda, variazioni della frequenza e del voltaggio, ecc.).
Mullen e Onsongo hanno elaborato un sistema multi-agente utilizzato per reagire a problemi di abbassamento della frequenza nelle reti elettriche [6]; tali problemi si verificano generalmente quando uno dei generatori smette di funzionare e non c'è quindi più abbastanza potenza per tutti i carichi. Il sistema presentato permette di selezionare in maniera intelligente i carichi da disattivare per riportare i parametri della rete ai loro valori nominali.
Un problema simile è stato preso in considerazione da Solanki, Khushalani Solanki e Schulz [7]. Il loro articolo propone un sistema multi-agente per la riconfigurazione della rete elettrica successivamente a un malfunzionamento; i carichi da disattivare vengono scelti in base a delle priorità e cercando di eliminare il fenomeno dell'isolamento (parti di rete che si ritrovano disconnesse dalla rete principale e prive di generatori).
Yu, Schulz e Srivastava propongono un sistema multi-agente simile applicato a navi militari con l'obiettivo di isolare la parte della rete guasta in modo che non ci siano discontinuità di funzionamento nel resto della rete.

2.3 Modello proposto

Molti dei lavori pubblicati sono estremamente specifici e si applicano a reti ben definite, ad esempio a una rete di abitazioni a livello nazionale [1]. In molti casi i modelli sopra citati reagiscono ad eventuali malfunzionamenti della rete e a perdite di potenza scollegando carichi secondo un certo criterio fino a riportare il consumo di potenza nella rete a livelli stabili [6].

Il modello valutato in questa tesi parte invece dal presupposto che sia necessario distribuire corrente elettrica all'interno di una rete secondo criteri ben precisi e indipendentemente dal tipo di rete stessa. Si potrebbe quindi modellare una rete molto estesa, ad esempio una

fabbrica, oppure una rete composta solo da pochi nodi, come un'abitazione in cui è necessario distribuire corrente a un numero limitato di carichi.
Non si fa riferimento, quindi, a casi particolari come prevedere l'utilizzo di corrente da parte delle abitazioni di un Paese ma si è deciso di astrarre dal tipo di rete modellata.
Un'altra importante differenza è che è previsto che i carichi possano lavorare a diversi livelli di potenza, il che permette di reagire a eventuali guasti diminuendo la potenza fornita a uno o più carichi senza essere costretti a disattivarli completamente.

I requisiti del modello sono i seguenti:

1. Il modello deve essere flessibile in modo da adattarsi a più reti possibili, indipendentemente dalla loro topologia e dai carichi e generatori che la popolano.
2. Il modello deve permettere la creazione di una gerarchia tra i nodi della rete in modo da poter dare la priorità ai nodi considerati fondamentali a scapito di quelli meno importanti.
3. Il modello deve tenere conto del fatto che alcuni nodi hanno più di due stati (acceso e spento); alcuni nodi, infatti, potrebbero assorbire una potenza variabile che dipende dalla situazione. Ad esempio un frigorifero potrebbe consumare più o meno potenza a seconda della temperatura che va mantenuta o della velocità con cui si deve raggiungere tale temperatura.
4. Il modello ha come obiettivo la distribuzione della potenza all'interno della rete: ogni agente deve ricevere la massima potenza disponibile compatibilmente con la propria priorità e con la potenza massima fornita dai generatori.
5. Il modello deve essere in grado di trovare la soluzione ottima al problema e non solo una delle possibili soluzioni. Nel nostro caso ciò significa che data una potenza massima disponibile è necessario distribuire l'energia ai nodi in modo che ogni nodo riceva più corrente possibile, compatibilmente con la gerarchia sopra citata.
6. Il modello deve poter gestire situazioni mutevoli: ciò significa che una modifica della rete (carichi presenti, potenza disponibile, ecc.) deve generare una nuova soluzione al problema. Questo include anche i transitori: un nodo potrebbe consumare una certa quantità di potenza durante l'accensione e poi una quantità diversa di potenza durante il normale funzionamento.

Il modello valutato in questo lavoro è stato sviluppato da Tartara [8] e successivamente modificato dall'autore della tesi così da eliminare alcuni problemi che si potevano presentare in situazioni particolari.

2.4 Implementazione del modello

Dati i requisiti sopra elencati l'autore del modello ha deciso di risolvere il problema tramite un sistema multi-agente cooperativo. In un sistema multi-agente cooperativo gli agenti collaborano tra di loro per trovare la soluzione ad un problema (in questo caso per spartirsi una risorsa disponibile in quantità limitata); questo è in contrasto con i sistemi multi-agente competitivi in cui gli agenti competono tra di loro per fornire o acquisire una certa risorsa.
Entrambi i tipi di sistemi multi-agente sono stati utilizzati per gestire le reti elettriche; ad esempio [2] e [3] sono sistemi multi-agente competitivi, mentre [1], [4], [5], [6] e [7] sono sistemi multi-agente cooperativi.
Il modello presentato in questo lavoro è basato sui DCOP, Distributed Constraint Optimization Problem [9].
Un DCOP è un problema in cui si cerca di minimizzare (o massimizzare) una funzione di costo in presenza di vincoli utilizzando un sistema distribuito multi-agente. Più in dettaglio un DCOP è definito da:

1. Un insieme di agenti.
2. Un insieme di variabili.
3. Un insieme di domini associati alle variabili; ogni dominio deve essere finito.
4. Un insieme di funzioni che associano un costo ai vincoli tra variabili.
5. Una funzione che associa le variabili agli agenti.
6. Un operatore che aggrega tutti i costi, solitamente tale operatore è la sommatoria.

La scelta di utilizzare un DCOP per modellare la gestione di una rete elettrica dipende principalmente dal fatto che i DCOP sono un problema noto visto che sono stati studiati per decenni e sono dunque ampiamente trattati in letteratura; ciò permette di gestire una rete elettrica senza dover pensare a una soluzione *ad hoc* ma utilizzando invece uno dei tanti algoritmi che sono stati creati per risolvere i DCOP.
I DCOP, inoltre, sono per loro natura distribuiti: ciò permette a tutti i nodi della rete di contribuire alla soluzione del problema e quindi evita i noti problemi che affliggono i sistemi centralizzati. In primo luogo non esiste un singolo point of failure, il che permette alla rete di funzionare anche in caso di guasto di alcuni suoi componenti.
In secondo luogo si migliorano le prestazioni: la gestione di una rete elettrica è un problema computazionalmente complesso e la scelta di un sistema distribuito permette di ridurre i

problemi prestazionali di un sistema centralizzato.
Tutti gli esperimenti presentati in questa tesi sono stati però condotti su una singola macchina; ulteriori dettagli su come si è simulato un sistema distribuito su un singolo computer sono disponibili nel Capitolo 4.

2.5 Applicazioni del modello

Come già accennato il modello deve poter essere usato per gestire una rete elettrica qualsiasi, indipendentemente dalla topologia della rete stessa. Verranno ora illustrate alcune possibili applicazioni del modello.

Casa non allacciata alla rete elettrica nazionale

In questo caso la potenza elettrica viene fornita da fonti rinnovabili e da batterie; in generale le batterie fungono da sistema di backup nel caso di guasti o di problemi nella generazione dell'energia dovuti al tempo atmosferico. Questa rete presenta una doppia sfida in quanto è necessario gestire non solo i carichi ma anche i generatori.
Per quanto riguarda i carichi potrebbe essere necessario diminuire la loro potenza di funzionamento in modo da non consumare troppo velocemente la batteria e in alcuni casi i carichi a bassa priorità devono essere completamente spenti.
Per quanto riguarda i generatori è necessario decidere quando la potenza fornita dai generatori che utilizzano fonti rinnovabili è insufficiente: non appena ciò accade bisogna iniziare ad utilizzare la batteria.
Un sistema leggermente più complesso permetterebbe ai due tipi di generatori di funzionare contemporaneamente, utilizzando la batteria per integrare gli altri generatori. Anche in questo caso, però, bisognerebbe stare attenti a evitare un uso eccessivo della batteria che potrebbe portare velocemente a una situazione in cui tutti i generatori della rete non sono in grado di fornire una quantità sufficiente di energia ai carichi.

Nave

Su una nave l'energia elettrica è prodotta da generatori azionati dai motori. In generale il numero di carichi presenti è elevato, il che significa che non è possibile soddisfare il fabbisogno di ogni carico. In questa situazione si rende necessario decidere come gestire l'energia presente in base alle priorità associate ad ogni carico.

Alcuni carichi, pur avendo bassa priorità, devono ricevere energia elettrica di tanto in tanto; per evitare il fenomeno della *starvation* si sfrutta il fatto che il modello è in grado di gestire reti dinamiche: ad intervalli regolari le priorità dei carichi possono essere alterate per modificare il modo con cui l'energia viene distribuita nella rete.

Aereo

Nell'aviazione, in particolare in quella civile, la sicurezza assume un'importanza fondamentale. Considerando quanto l'elettronica ha cambiato i velivoli e l'importanza che ha assunto ciò significa che la corretta distribuzione di corrente all'interno della rete elettrica di un velivolo è molto importante.

Su un aereo, così come su una nave, la potenza elettrica è normalmente fornita da generatori azionati dai motori; i velivoli hanno però sono più complicati perché hanno ulteriori generatori utilizzabili in caso di emergenza.

Esempi di generatori di emergenza sono:

1. Le batterie: ogni aereo è dotato di una o più batterie utilizzabili nel caso in cui tutti gli altri generatori siano guasti. Questi generatori forniscono energia per un periodo di tempo limitato e a causa delle loro caratteristiche sono in grado di fornire energia solo ad alcuni sistemi del velivolo.
2. APU: gli *auxiliary power unit* sono piccole turbine a gas; in generale il loro compito è fornire energia elettrica quando i motori sono spenti e permettere l'accensione dei motori stessi. Un loro utilizzo secondario è quello di generatore di emergenza nel caso di guasto degli alternatori principali.
3. Ram air turbine: queste turbine alloggiate all'interno del velivolo vengono automaticamente estratte in caso di guasto. La produzione di energia avviene grazie al flusso d'aria che investe il velivolo, pertanto la quantità di energia generata dipende dalla velocità del velivolo.

Questa breve spiegazione dei generatori presenti su un velivolo dimostra come la gestione della corrente elettrica sia estremamente importante. In particolare, in caso di guasto l'equipaggio potrebbe essere troppo impegnato a tenere l'aereo in volo per potersi occupare di tutti i dettagli riguardanti i sistemi elettrici di emergenza. In casi come questo un sistema automatico di gestione dell'energia può aiutare l'equipaggio e rendere più sicuro il volo.

La gestione dell'energia su un velivolo è sicuramente un problema complesso da risolvere ma le caratteristiche del modello qui preso in considerazione sembrano ben adattarsi al problema;

in particolare la capacità del modello di gestire reti dinamiche è importante perché la rete elettrica di un aereo in caso di guasto cambia velocemente, sia per quanto riguarda le priorità associate ai carichi che per la potenza fornita dai generatori.

Capitolo 3

Nel capitolo precedente sono state descritte alcune caratteristiche che il modello deve possedere. Il modello verrà ora descritto dettagliatamente e verrà dimostrato che possiede tutte queste caratteristiche.
La Sezione 3.1 spiega come è composto il DCOP utilizzato dal modello.
La Sezione 3.2 mostra una semplice classificazione dei vincoli di un DCOP; tale classificazione è necessaria per comprendere il funzionamento del programma usato per risolvere i DCOP, analizzato nel Capitolo 4.
La Sezione 3.3 analizza le funzioni di costo che associano i costi ai vincoli nel DCOP.

3.1 Il DCOP usato dal modello

Prima di illustrare il DCOP è necessario spiegare quali sono i parametri che il modello utilizza per descrivere una rete elettrica.

- Potenza disponibile: questo parametro indica la potenza che i generatori forniscono alla rete. Il modello deve ovviamente far sì che la potenza istantanea consumata da tutti i carichi non superi la potenza massima disponibile.
- Carichi: ad ogni carico presente nella rete è associato un insieme contenente tutte le potenze istantanee che quel carico può assorbire.
- Gerarchia delle priorità: ad ogni carico è associata una priorità; tale sistema permette di organizzare i carichi in base a una gerarchia basata sull'importanza di ogni carico.

Come detto nel capitolo precedente un DCOP può essere descritto dalla seguente tupla: insieme di agenti, insieme di variabili, insieme dei domini delle variabili, rapporto tra agenti e variabili, insieme delle funzioni di costo associate ai vincoli e operatore di aggregazione dei costi.

- Insieme degli agenti: ogni nodo della rete (cioè ogni carico) viene modellato con un agente. Tale agente ha il compito di collaborare con tutti gli altri agenti alla ricerca della soluzione del problema; una volta trovata la soluzione l'agente comunica al controller del carico la potenza da utilizzare.

- Insieme delle variabili: ad ogni carico viene associata una variabile che consiste nella potenza attualmente utilizzata dal carico.
- Rapporto tra variabili e agenti: la funzione che associa agenti e variabili è biunivoca, cioè ad ogni agente viene associata una sola variabile e viceversa. Sebbene i DCOP permettano di associare più variabili allo stesso agente, nel modello qui presentato non si fa uso di tale possibilità. Data la corrispondenza tra variabili e agenti è possibile considerare questi due termini sinonimi nel seguito di questa trattazione.
- Insieme dei domini: ad ogni variabile è associato un dominio finito. Negli esperimenti che verranno mostrati in seguito tutti gli agenti hanno lo stesso dominio per motivi di semplicità ma il modello permette comunque di associare a ogni variabile un dominio diverso.
- Insieme delle funzioni di costo: le funzioni di costo verranno trattate nella sezione 3.
- Operatore di aggregazione dei costi: tutti i costi associati ai vincoli vengono sommati; l'obiettivo dell'algoritmo è la minimizzazione della somma dei costi.

Prima di illustrare le funzioni di costo usate dal modello è necessario introdurre una classificazione dei vincoli. Questa non è una classificazione generale relativa ai DCOP ma è una classificazione utilizzata dal programma che è stato usato per seguire gli esperimenti.

3.2 Classificazione dei vincoli

Classificazione in base al costo associato

I vincoli *hard* sono associati a un costo booleano: questo tipo di vincoli o è soddisfatto oppure non lo è; non esiste la possibilità di soddisfare parzialmente un vincolo *hard.*
I vincoli *soft* sono associati a un numero che indica il grado di soddisfacimento del vincolo. È ovviamente possibile utilizzare un vincolo *soft* per modellarne uno *hard*: in questo caso al vincolo viene associato il valore zero quando il vincolo è soddisfatto e il valore infinito quando il vincolo non è soddisfatto.

Classificazione in base al calcolo della funzione di costo

I vincoli *intensionali* sono vincoli associati a una funzione; tale funzione permette di calcolare il grado di soddisfacimento del vincolo in base ai valori assunti dalle variabili.
Un esempio di vincolo intensionale è il seguente:

$costo = \frac{|(x_1 - x_2)|}{x_3}$, dove x_1 , x_2 e x_3 sono le variabili coinvolte nel vincolo.

I vincoli *estensionali* sono vincoli in cui la funzione di costo si limita ad associare una o più tuple di valori assunti dalle variabili a un numero che rappresenta il costo associato al vincolo. Il vincolo è definito estensionale perché è necessario esplicitare tutte le tuple di valori che possono essere assunti dagli agenti coinvolti nel vincolo e ad ogni tupla (o gruppo di tuple) si associa un costo.
Un esempio di vincolo estensionale è il seguente:

0: 1 1|2 2|3 3
1: 2 1|2 3|3 1
2: 1 2|3 2|3 0

I numeri nella prima colonna sono i costi, mentre le tuple separate dal simbolo '|' indicano le tuple a cui è associato quel costo. Ad esempio la tupla “2 1” è associata al costo 1; ciò significa che quando la prima variabile del vincolo assume valore 2 e la seconda variabile assume valore 1 il costo associato al vincolo è 1.

Nel modello qui valutato sono presenti sia vincoli hard che vincoli soft; inoltre tutti i vincoli sono intensionali.

3.3 Funzioni di costo

Il modello prevede l'esistenza di un vincolo per ogni coppia di agenti i e j con i <= j; la funzione di costo associata al vincolo è:

$$\frac{(X_i - x_i) + (X_j - x_j)}{(X_i + X_j)} \quad \text{se} \quad p_i = p_j$$

$$\frac{(X_j - x_j)}{X_j} \quad \text{se} \quad p_i > p_j \wedge (x_i = X_i \vee x_j = 0)$$

$$\infty \; altrimenti$$

Inoltre bisogna rispettare il vincolo imposto dalla potenza massima disponibile.

$$\sum_i x_i \leq P_{max}$$

In queste formule p_n è la priorità dell'agente n, x_n è la potenza usata dall'agente n (cioè il valore della variabile associata all'agente n), X_n è la potenza massima utilizzabile dall'agente n (cioè il massimo del dominio della variabile associata all'agente n) e P_{max} è la potenza massima disponibile nella rete.
Il modello prevede che le variabili e i costi assumano valori reali mentre le priorità sono degli interi nell'intervallo [1, p] dove 1 è la priorità minima e p la priorità massima.
La prima formula viene utilizzata per ogni coppia di agenti che hanno la stessa priorità e prevede che il costo sia direttamente proporzionale alla somma degli scostamenti delle due variabili dal loro valore massimo. Il denominatore serve a pesare il costo facendo sì che non assuma valori superiori ad uno; in questo modo è più semplice confrontare i costi tra agenti che hanno domini molto diversi tra loro.
La seconda formula viene utilizzata per ogni coppia di agenti che hanno diverse priorità e prevede che il costo sia direttamente proporzionale allo scostamento della variabile associata all'agente a bassa priorità rispetto al suo valore massimo. Si noti che la formula si applica solo se l'agente a priorità alta sta funzionando a massima potenza oppure se l'agente a bassa priorità è spento (nel qual caso il costo associato al vincolo è banalmente 1, cioè il valore

massimo). Grazie a questa condizione è possibile costringere il sistema a fornire tutta la potenza disponibile agli agenti ad alta priorità, lasciando agli agenti a bassa priorità eventuali rimanenze.
Si noti che le funzioni di costo utilizzate dal modello sono progettate in maniera tale da generare costi che sono sempre compresi nell'intervallo [0, 1].

L'ultima formula è associata a un vincolo hard che è anche l'unico vincolo n-ario del modello; tale vincolo impone che la potenza consumata dai carichi non superi la potenza massima disponibile nella rete.

Il modello sopra presentato permette di creare una gerarchia di agenti tramite le priorità: gli agenti a priorità elevata, infatti, sono privilegiati in quanto ottengono più facilmente energia; gli agenti a bassa priorità, invece, ricevono potenza solo dopo che sono stati soddisfatte le richieste degli agenti ad alta priorità.
Dato il modello di cui sopra non è mai possibile che un agente a bassa priorità riceva potenza se gli agenti ad alta priorità non stanno tutti funzionando a potenza massima.

Il modello descritto permette di risolvere problemi "a tempo costante": ogni volta che scatta un timer il problema viene risolto. Ciò significa che eventuali cambiamenti al problema (modifiche nei domini o nella potenza massima disponibile, aggiunta o rimozione di carichi, ecc.) portano a cambiamenti nella soluzione, che deve essere ricalcolata. Tale situazione è mostrata nella Figura 3.1.

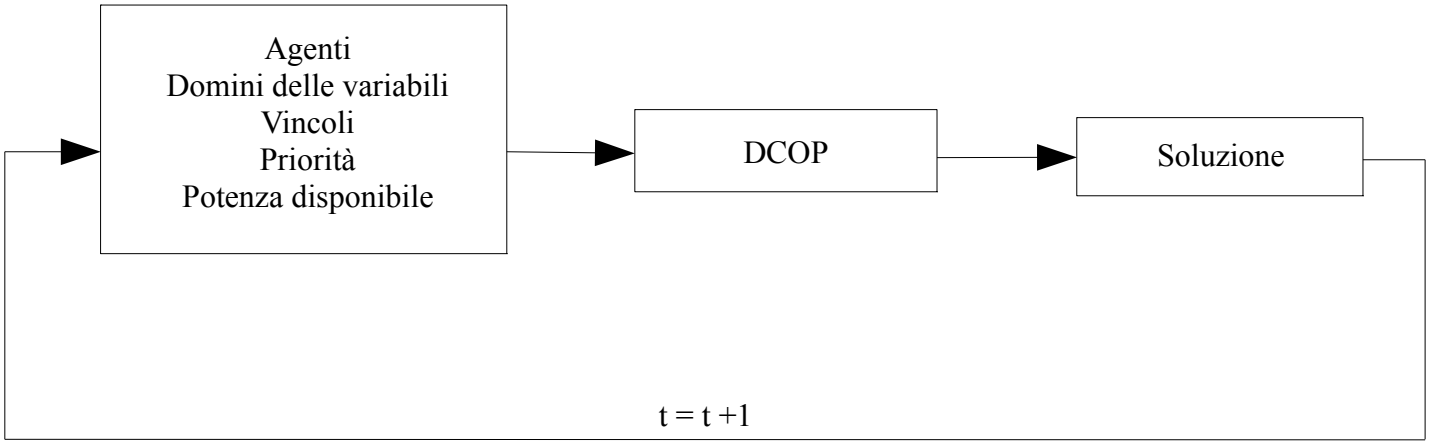

Figura 3.1

La Figura 3.1 mostra come ad ogni evento il DCOP viene risolto con i nuovi parametri che caratterizzano la rete.

Si noti che, come specificato nella figura, la soluzione trovata al tempo *t* può essere considerata un parametro da utilizzare per il calcolo della soluzione al tempo *t+1;* ciò può essere utile se si vuole mantenere la soluzione il più possibile stabile. Per motivi tecnici non è stato possibile sfruttare questa capacità del modello.

Capitolo 4

In questo capitolo si dà una breve descrizione di FRODO, lo strumento usato per risolvere i DCOP utilizzati per valutare il modello. I contenuti di questo capitolo forniscono solo una breve panoramica di FRODO, per ulteriori informazioni si rimanda al relativo manuale [10].
Nella Sezione 4.1 si forniscono informazioni generali sullo strumento.
La Sezione 4.2 specifica le modalità di funzionamento del programma.
La Sezione 4.3 illustra i parser utilizzabili da FRODO e i relativi pregi e difetti.
La Sezione 4.4 descrive le metriche che FRODO offre per valutare le prestazioni degli algoritmi che risolvono il DCOP.
La Sezione 4.5 classifica gli algoritmi forniti da FRODO in base al loro modo di funzionamento.
La Sezione 4.6 illustra alcune limitazioni di FRODO e le modifiche al modello che si sono rese necessarie per aggirare tali limitazioni.
La Sezione 4.7 mostra e commenta un file utilizzato per specificare un DCOP in FRODO.

4.1 Informazioni generali su FRODO

Lo strumento usato per risolvere i problemi relativi al modello è FRODO (http://frodo2.sourceforge.net). FRODO è un framework open-source scritto in Java utilizzabile per risolvere DCOP; uno dei suoi punti di forza è il fatto che sono disponibili vari algoritmi per risolvere i DCOP e che i file utilizzati per descrivere i problemi non dipendono dagli algoritmi. Ciò rende molto più semplice comparare le prestazioni degli algoritmi utilizzati per valutare il modello.
In questo lavoro FRODO è stato utilizzato come programma stand-alone ma è anche possibile sfruttare le API fornite per integrare FRODO con un programma pre-esistente.
Il fatto che sia scritto in Java permette a FRODO di essere utilizzato su molte diverse piattaforme senza modifiche e ad ogni modo essendo software open-source (pubblicato con licenza GNU Affero General Public License versione 3) può essere liberamente modificato in base alle proprie esigenze.

4.2 Modalità di funzionamento

FRODO è in grado di funzionare sia in modalità centralizzata che in modalità distribuita.
Nel primo caso gli agenti vengono implementati da FRODO con dei thread Java appartenenti allo stesso processo; siccome i thread condividono lo stesso spazio di indirizzamento lo scambio di messaggi avviene tramite un semplice scambio di puntatori.
Per far sì che sia possibile misurare il tempo necessario per calcolare la soluzione, FRODO utilizza un *central mailer*; tale componente riceve tutti i messaggi inviati dagli agenti e li consegna, uno alla volta e in ordine di *timestamp*, ai destinatari. Ad ogni agente è assegnato un *clock* che viene fatto partire alla ricezione del messaggio e viene fermato quando l'agente ha terminato di processare il messaggio. In questo modo al termine della computazione è possibile ottenere una stima del tempo necessario per il calcolo della soluzione.
Nella modalità distribuita ogni agente risiede su un computer diverso ed è implementato come un demone; ogni demone attende che un controller centrale comunichi il problema da risolvere e successivamente comunica con gli altri demoni tramite il protocollo TCP; questa modalità può essere utilizzata anche su una singola macchina a patto che ogni agente abbia a disposizione un processore (o un core).
Tutti gli esperimenti qui presentati sono stati eseguiti in modalità centralizzata.

4.3 Parser utilizzabili da FRODO

FRODO è in grado di utilizzare due diversi parser che leggono il file XML contenente il problema da risolvere e lo elaborano in modo da poterlo poi inviare in input all'algoritmo scelto. Il primo parser è incluso nella distribuzione di FRODO e permette di risolvere problemi contenenti vincoli estensionali soft; più in particolare il parser predefinito è in grado di comprendere un sottoinsieme del formato XCSP 2.1 [11] che è un formato XML utilizzato per descrivere reti di vincoli.
L'altro parser utilizzabile da FRODO è JaCoP (http://www.jacop.eu); tale parser, a differenza del precedente, è in grado di comprendere un sovrainsieme di XCSP e in particolare è in grado di gestire tutti i tipi di vincoli: intensionali ed estensionali, *hard* e *soft*.
Date le sue potenzialità il parser JaCoP è stato scelto per eseguire tutti gli esperimenti presentati in questo lavoro. Sfortunatamente questo parser non è in grado di gestire numeri reali, quindi tutti i domini devono contenere solo valori interi e che tutti i costi devono essere

interi; ciò significa che è stato necessario introdurre una modifica al modello presentato nel capitolo precedente che invece si basava sui numeri reali.
La modifica consiste nel moltiplicare il numeratore di tutte le funzioni di costo per una potenza del dieci (un parametro di scala). Siccome tutti i costi sono moltiplicati per lo stesso numero la soluzione del problema non cambia, semplicemente anziché avere costi nell'intervallo [0, 1] i costi valutati da FRODO sono nell'intervallo [0, s], dove s è il parametro di scala.
Il parser JaCoP ha anche il vantaggio di gestire vincoli hard basati su somme pesate; ciò è stato utile perché il vincolo n-ario presente nel modello è facilmente modellabile come una somma pesata: tutte i valori assunti dalle variabili sono pesati con il coefficiente 1 (perché tutti hanno la stessa importanza e dunque lo stesso peso) e poi sommati.

4.4 Metriche di valutazione degli algoritmi

FRODO fornisce varie metriche utilizzabili per valutare la soluzione di un problema e le prestazioni ottenute dall'algoritmo scelto:

1. Valori assunti dalle variabili: per ogni variabile presente nel problema viene mostrato il valore che la variabile assume nella soluzione.
2. Costo della soluzione: il costo totale viene utilizzato per valutare la bontà della soluzione; se FRODO non è stato in grado di trovare una soluzione valida il costo totale viene riportato come infinito.
3. Tempo simulato: il tempo simulato è una metrica che permette di valutare le prestazioni di un algoritmo quando si usa la modalità di esecuzione centralizzata. L'obiettivo di questo parametro è mostrare quanto tempo verrebbe impiegato per calcolare la soluzione se si utilizzasse una piattaforma veramente distribuita.
4. Messaggi scambiati: mostra un rendiconto contenente quanti e quali messaggi sono stati scambiati; vengono mostrati sia il numero di messaggi che la quantità di informazioni scambiate (in byte).
5. Alberi creati dall'algoritmo: ogni algoritmo mostra una serie di alberi che sono stati creati per risolvere il problema; la quantità e il tipo di alberi dipende dall'algoritmo stesso.

4.5 Classificazione degli algoritmi offerti da FRODO

Come sopra accennato uno dei punti di forza di FRODO è il fatto che sono stati implementati molti algoritmi per risolvere DCOP; gli algoritmi presenti possono essere così catalogati:

1. Algoritmi a ricerca globale: questi algoritmi garantiscono che, se esiste una soluzione, questa sarà trovata; inoltre se esistono più soluzioni verrà sicuramente scelta quella ottima, cioè quella che minimizza (o massimizza) il costo totale. In pratica ciò significa che questi algoritmi esplorano tutto lo spazio delle possibili soluzioni e forniscono in output la soluzione migliore. Queste importanti proprietà hanno però delle ripercussioni sulla complessità spaziale e temporale dell'algoritmo stesso. Esempi di algoritmi a ricerca globale sono ADOPT [12], DPOP [13], ASO-DPOP [14], MB-DPOP [15], O-DPOP [16] e SynchBB [17].
2. Algoritmi a ricerca locale: questi algoritmi esplorano solo una piccola parte dello spazio delle soluzioni; a partire da un assegnamento iniziale (solitamente casuale) questi algoritmi si spostano verso altri assegnamenti finché non trovano la soluzione localmente ottima o scade il tempo a loro disposizione. Questi algoritmi hanno un tempo di esecuzione molto più basso rispetto agli algoritmi a ricerca globale ma non garantiscono di trovare una soluzione o che la soluzione trovata sia la soluzione ottima. Esempi di algoritmi a ricerca locale sono DSA [18], MGM [19], MGM-2 [19] e Max-Sum [20].
3. Algoritmi con privacy: questi algoritmi, che possono essere sia a ricerca locale che a ricerca globale, si differenziano dagli altri perché permettono agli agenti di nascondere il valore della loro variabile e il dominio ad essa associata; ciò significa che gli agenti devono da un lato cooperare per trovare la soluzione e dall'altro nascondere informazioni agli altri agenti. Date le tecniche usate da questi algoritmi (riempimento dei domini con valori casuali "di disturbo", cifratura, ecc.) le prestazioni sono piuttosto scarse sia dal punto di vista temporale che da quello spaziale. Esempi di questo tipo di algoritmi sono MPC-DisCSP4 [21], MPC-DisWCSP4 [22], P-DPOP [23], P^2-DPOP [24] e $P^{3/2}$-DPOP [24].

4.6 Limitazioni di FRODO e loro impatto sul modello

FRODO ha alcune limitazioni che hanno avuto un impatto sugli esperimenti eseguiti; la prima limitazione risiede nell'impossibilità del parser JaCoP di gestire i numeri reali e ciò ha creato la necessità di introdurre il parametro di scala sopra descritto.
La seconda limitazione sta nel fatto che all'interno del costrutto *if-then-else* offerto da JaCoP non è possibile ritornare un costo infinito; tale costrutto è necessario per modellare la funzione di costo associata ai vincoli che coinvolgono agenti con diverse priorità.
Per convenienza viene qui riportata la formula:

$$\frac{(X_j - x_j)}{X_j} \quad \text{se} \quad p_i > p_j \wedge (x_i = X_i \vee x_j = 0)$$

Sono state valutate due possibili soluzioni a questo inconveniente:

1. Il vincolo viene spezzato in due: un vincolo hard contenente solo la condizione e un vincolo soft il cui costo è rappresentato dalla formula del ramo *then*. Questa soluzione funziona perché tutti i vincoli hard devono essere soddisfatti affinché la soluzione sia valida.
2. Il vincolo viene modificato e nel ramo *else* ritorna un numero intero invece che infinito. In questo caso bisogna stare attenti a far sì che il costo sia sufficientemente alto da poter essere considerato infinito. Il ragionamento che è stato utilizzato per trovare questo valore è il seguente: ogni vincolo presente nel problema (a parte il vincolo *hard* n-ario) ha un costo che varia nell'intervallo [0, s], dove s è il parametro di scala. Ciò significa che il costo massimo assumibile da una soluzione è pari al numero di vincoli del problema moltiplicato per il parametro di scala. Dunque un qualunque valore superiore a questo può essere usato per approssimare infinito nel ramo *else*.

In questo lavoro si è scelto di usare la seconda soluzione così da non introdurre vincoli che non fossero originariamente previsti nel modello.

4.7 Formato dei file contenenti DCOP risolvibili da FRODO

Segue un esempio di file XML che rappresenta un problema risolvibile con FRODO:

```
<?xml version="1.0" encoding="UTF-8"?>
<instance>
  <presentation name="exp_4_2_10_3Pdiv4" maxConstraintArity="4"
maximize="false" format="XCSP 2.1_FRODO" />
  <domains nbDomains="1">
    <domain name="domain" nbValues="11">0 1 2 3 4 5 6 7 8 9 10</domain>
  </domains>
  <agents nbAgents="4">
    <agent name="agent_1" />
    <agent name="agent_2" />
    <agent name="agent_3" />
    <agent name="agent_4" />
  </agents>
  <variables nbVariables="4">
    <variable name="variable_1" domain="domain" agent="agent_1" />
  <variable name="variable_2" domain="domain" agent="agent_2" />
    <variable name="variable_3" domain="domain" agent="agent_3" />
    <variable name="variable_4" domain="domain" agent="agent_4" />
  </variables>
  <functions nbFunctions="2">
    <function name="same_priority" return="int">
      <parameters>int x1 int X1 int x2 int X2</parameters>
      <expression>

<functional>div(mul(add(sub(X1,x1),sub(X2,x2)),100),add(X1,X2))</functional
>
      </expression>
    </function>
    <function name="different_priority" return="int">
      <parameters>int x1 int X1 int x2 int X2</parameters>
      <expression>
        <functional>if(or(eq(x1,X1),eq(x2,0)), div(mul(sub(X2,x2),100),X2),
601)</functional>
      </expression>
    </function>
  </functions>
```

```
  <constraints nbConstraints="7">
    <constraint name="max_power" arity="4" scope="variable_1 variable_2
variable_3 variable_4" reference="global:weightedSum">
      <parameters>
        [ { 1 variable_1 } { 1 variable_2 } { 1 variable_3 } { 1 variable_4
} ]
        <le />
        30
      </parameters>
    </constraint>
    <constraint name="variable_1_variable_2" arity="2" scope="variable_1
variable_2" reference="same_priority">
      <parameters>variable_1 10 variable_2 10</parameters>
    </constraint>
    <constraint name="variable_1_variable_3" arity="2" scope="variable_1
variable_3" reference="different_priority">
      <parameters>variable_1 10 variable_3 10</parameters>
    </constraint>
    <constraint name="variable_1_variable_4" arity="2" scope="variable_1
variable_4" reference="different_priority">
      <parameters>variable_1 10 variable_4 10</parameters>
    </constraint>
    <constraint name="variable_2_variable_3" arity="2" scope="variable_2
variable_3" reference="different_priority">
      <parameters>variable_2 10 variable_3 10</parameters>
    </constraint>
    <constraint name="variable_2_variable_4" arity="2" scope="variable_2
variable_4" reference="different_priority">
      <parameters>variable_2 10 variable_4 10</parameters>
    </constraint>
    <constraint name="variable_3_variable_4" arity="2" scope="variable_3
variable_4" reference="same_priority">
      <parameters>variable_3 10 variable_4 10</parameters>
    </constraint>
  </constraints>
</instance>
```

Le varie parti di questo file verranno ora analizzate:

```
<presentation name="exp_4_2_10_3Pdiv4" maxConstraintArity="4"
```

```
maximize="false" format="XCSP 2.1_FRODO" />
```

Questo tag contiene informazioni generali sul problema da risolvere: nome (non obbligatorio), massima arità dei vincoli e operazione da effettuare (minimizzare o massimizzare i costi).

```
<domains nbDomains="1">
      <domain name="domain" nbValues="11">0 1 2 3 4 5 6 7 8 9 10</domain>
</domains>
```

Questo blocco contiene tutti i domini delle variabili; come già accennato negli esperimenti qui presentati tutte la variabili hanno lo stesso dominio.

```
<agents nbAgents="4">
    <agent name="agent_1" />
    <agent name="agent_2" />
    <agent name="agent_3" />
    <agent name="agent_4" />
  </agents>

  <variables nbVariables="4">
    <variable name="variable_1" domain="domain" agent="agent_1" />
    <variable name="variable_2" domain="domain" agent="agent_2" />
    <variable name="variable_3" domain="domain" agent="agent_3" />
    <variable name="variable_4" domain="domain" agent="agent_4" />
  </variables>
```

Questi due blocchi definiscono gli agenti e le variabili che compongono il problema; nel secondo blocco ogni variabile viene assegnata a uno degli agenti (in questo caso la relazione è biunivoca).

```
 <functions nbFunctions="2">
    <function name="same_priority" return="int">
      <parameters>int x1 int X1 int x2 int X2</parameters>
      <expression>

<functional>div(mul(add(sub(X1,x1),sub(X2,x2)),100),add(X1,X2))</functional
>
      </expression>
```

```
    </function>
    <function name="different_priority" return="int">
      <parameters>int x1 int X1 int x2 int X2</parameters>
      <expression>
        <functional>if(or(eq(x1,X1),eq(x2,0)), div(mul(sub(X2,x2),100),X2),
601)</functional>
      </expression>
    </function>
  </functions>
```

In questo blocco vengono definiti i vincoli soft del problema; eventuali vincoli hard sarebbero definiti in un blocco "*predicates*".
Ogni funzione di costo dichiara innanzitutto le variabili usate e il loro tipo (per ora *int* è l'unico tipo accettato), successivamente viene esplicitata la funzione stessa. Ogni funzione deve necessariamente ritornare un valore intero, mentre nel caso dei predicati ogni predicato deve ritornare un valore tra *true* e *false*.
La prima delle funzioni sopra mostrate viene usata dagli agenti che hanno la stessa priorità mentre le seconda è per gli agenti con diverse priorità.

```
   <constraint name="max_power" arity="4" scope="variable_1 variable_2
variable_3 variable_4" reference="global:weightedSum">
      <parameters>
        [ { 1 variable_1 } { 1 variable_2 } { 1 variable_3 } { 1 variable_4
} ]
        <le />
        30
      </parameters> </constraint>
```

Il primo vincolo di ogni problema è il vincolo n-ario *hard* che stabilisce la potenza massima disponibile; come già accennato tale vincolo viene rappresentato da una somma pesata di tutte le variabili. Il tag *<le />* significa "lesser or equal", altri tag sono disponibili per eventuali altre applicazioni.

```
  <constraint name="variable_1_variable_2" arity="2" scope="variable_1
variable_2" reference="same_priority">
      <parameters>variable_1 10 variable_2 10</parameters>
    </constraint>
```

```
    <constraint name="variable_1_variable_3" arity="2" scope="variable_1
variable_3" reference="different_priority">
      <parameters>variable_1 10 variable_3 10</parameters>
    </constraint>
```

Questi due vincoli mostrano come utilizzare le funzioni sopra definite. Tutti i vincoli devono specificare un nome univoco, l'arità del vincolo, le variabili coinvolte e la funzione o il predicato a cui si riferiscono (nel caso di vincoli estensionali bisogna invece indicare la relazione di riferimento). Inoltre ogni vincolo deve poi specificare i parametri da passare alle funzioni (nel nostro caso le variabili coinvolte e delle costanti che rappresentano il massimo valore assumibile dalle variabili).

Per completare il quadro relativo a FRODO si mostra ora un esempio di problema e l'output fornito da FRODO.
Si supponga di avere un sistema multiagente così definito:

- 4 agenti, ognuno che controlla una diversa variabile
- Tutte le variabili hanno dominio [0...10]
- La potenza massima disponibile è 20
- Gli agenti sono organizzati in due classi di priorità; in particolare gli agenti 1 e 2 hanno priorità elevata mentre gli agenti 3 e 4 hanno priorità bassa.

Nel problema corrispondente il vincolo tra gli agenti 1 e 2 e il vincolo tra gli agenti 3 e 4 è associato alla seguente funzione di costo:

$$\frac{(X_i - x_i) + (X_j - x_j)}{(X_i + X_j)}$$

Questo è dovuto al fatto che gli agenti 1 e 2 appartengono alla stessa classe di priorità, e lo stesso accade per gli agenti 3 e 4.

I vincoli tra gli agenti 1 e 3, 1 e 4, 2 e 3, 2 e 4, invece, utilizzano la seguente funzione di costo perché coinvolgono agenti che appartengono a classi di priorità diverse:

$$\frac{(X_j - x_j)}{X_j} \quad \text{se} \quad x_i = X_i \vee x_j = 0$$

Quando il problema sopra definito viene risolto con uno degli algoritmi implementati da FRODO si ottiene il seguente output:

```
var `variable_4' = 0
var `variable_3' = 0
var `variable_2' = 10
var `variable_1' = 10
Total optimal cost: 300
Algorithm finished in 1,102 ms (simulated time)
Number of messages sent (by type):
      Backtrack:  1,463
      ELECT:      1,800
      NextVarChosen:    3
      NextVarProposal:  6
      NextVarRequest:   6
      Path: 1,463
      Solution:   3
      UB:   33
      VarOrderNoSpace:  3
      - Total:    4,780
Amount of information sent (by type, in bytes):
      Backtrack:  14,743
      ELECT:      29,763
      NextVarChosen:    279
      NextVarProposal:  996
      NextVarRequest:   378
      Path: 74,326
      Solution:   416
      UB:   2,331
      VarOrderNoSpace:  579
      - Total:    123,811
```

Si noti come sono presenti i seguenti dati:

1. Valori assegnati alle variabili

2. Costo della soluzione trovata
3. Tempo simulato (in ms)
4. Numero di messaggi scambiati (in totale e suddivisi per categorie)
5. Dimensione dei messaggi scambiati (in totale e suddivisi per categorie, misurati in byte)

Capitolo 5

In questo capitolo vengono analizzati i risultati degli esperimenti condotti.
La Sezione 5.1 spiega i parametri utilizzati per generare i problemi risolti dagli algoritmi.
Le sezioni seguenti illustrano le prestazioni degli algoritmi con alcuni dei problemi risolti; gli algoritmi sono valutati in base al tempo di esecuzione, ai messaggi scambiati e alla qualità della soluzione trovata.

5.1 Descrizione degli esperimenti eseguiti

Tutti i problemi risolti con FRODO hanno come obiettivo la massimizzazione degli agenti accesi; in altre parole l'obiettivo è di utilizzare tutta la potenza disponibile nella rete distribuendola agli agenti in base alla loro priorità. Eventuale potenza non utilizzata causa un aumento del costo della soluzione.

Per valutare il modello si è deciso di utilizzare i seguenti algoritmi:

- ADOPT [12]
- DPOP [13]
- MB-DPOP [15]
- MGM [19]
- SynchBB [17]

A parte MGM tutti gli algoritmi sopra citati sono algoritmi a ricerca globale. La scelta di privilegiare gli algoritmi a ricerca globale è dovuta al fatto che tali algoritmi garantiscono sempre di trovare la soluzione ottima, a patto che esista una soluzione valida al problema. Applicato al nostro caso ciò significa che la rete elettrica modellata dal DCOP utilizzerà sempre al meglio la potenza elettrica disponibile, senza sprechi.
L'algoritmo a ricerca locale è utile per verificare la differenza di prestazioni tra le due classi di algoritmi; gli esperimenti condotti sono inoltre mirati a comprendere quanto spesso MGM è in grado di trovare una soluzione e con che frequenza tale soluzione coincida con la soluzione ottima.

I seguenti parametri sono stati utilizzati per generare i problemi:

1. Numero agenti: 2, 4, 6, 8.
2. Numero classi di priorità: 1, 2, 3. Gli agenti sono sempre stati distribuiti il più uniformemente possibile sulle diverse classi di priorità.
3. Dominio delle variabili: il dominio delle variabili contiene sempre tutti gli interi nell'intervallo [0, d_max], dove per d_max sono stati scelti i valori 5, 10 e 15.
4. Potenza disponibile nella rete: 1, $\frac{1}{4}P$, $\frac{1}{2}P$, $\frac{3}{4}P$, P, dove P = d_max * numero agenti e d_max è il valore massimo del dominio. In sostanza P è la potenza che sarebbe necessaria per far funzionare tutti gli agenti a potenza massima.

Sono stati generati tutti i problemi ottenibili con tutte le possibili combinazioni dei parametri sopra illustrati, per un totale di 165 problemi.

È lecito aspettarsi che la complessità dei problemi aumenti all'aumentare del numero di agenti e della cardinalità del dominio; ciò è dovuto al fatto che la dimensione dello spazio delle soluzioni dipende proprio da questi due parametri. Nel caso in cui tutte le variabili abbiano lo stesso dominio, infatti, lo spazio delle soluzioni è popolato dalle combinazioni con ripetizione degli elementi del dominio presi n alla volta, dove n è il numero di agenti nel sistema.
Più complesso è stabilire quale impatto abbiano la potenza disponibile e le classi di priorità sulla complessità del problema. Se la potenza disponibile diminuisce il vincolo n-ario che regola la potenza massima utilizzabile diventa più selettivo e lo spazio delle soluzioni diventa più piccolo. Alla stessa maniera le priorità aggiungono vincoli che rendono più specifico il problema e più piccolo lo spazio delle soluzioni. Non è quindi possibile stabilire a priori l'impatto sulle prestazioni di questi parametri, ma è possibile supporre che algoritmi diversi reagiscano in maniera diversa.

Nelle prossime pagine sono presentati dei grafici che mostrano le prestazioni degli algoritmi con i problemi sopra descritti; segue una breve valutazione degli algoritmi.

Nei grafici d_max è il massimo valore del dominio, P è d_max * numero agenti, P_max è la potenza disponibile nella rete, pr è il numero di classi di priorità e n è il numero di agenti. Il tempo simulato è sempre espresso in ms e la dimensione dei messaggi in byte.

Le informazioni riguardanti i messaggi scambiati sono disponibili solo per i problemi contenenti 2 e 4 agenti in quanto registrare tutti i messaggi è computazionalmente molto costoso e quindi non è possibile con un numero più elevato di agenti.
Nel caso degli algoritmi a ricerca locale (nel nostro caso MGM) non c'è la certezza di riuscire a trovare una soluzione. I casi in cui MGM non è mai riuscito a trovare una soluzione sono segnalati nei grafici dall'assenza del valore relativo a MGM; in tutti gli altri casi i dati riguardanti MGM sono relativi alla soluzione migliore trovata. Tutti i problemi sono stati risolti con il timeout impostato a 60 minuti e nel caso di MGM il numero di cicli eseguiti dall'algoritmo è impostato a 200.
Si noti che nei seguenti diagrammi l'asse Y usa una scala logaritmica e non lineare.

5.2 Analisi delle prestazioni in base al numero di agenti

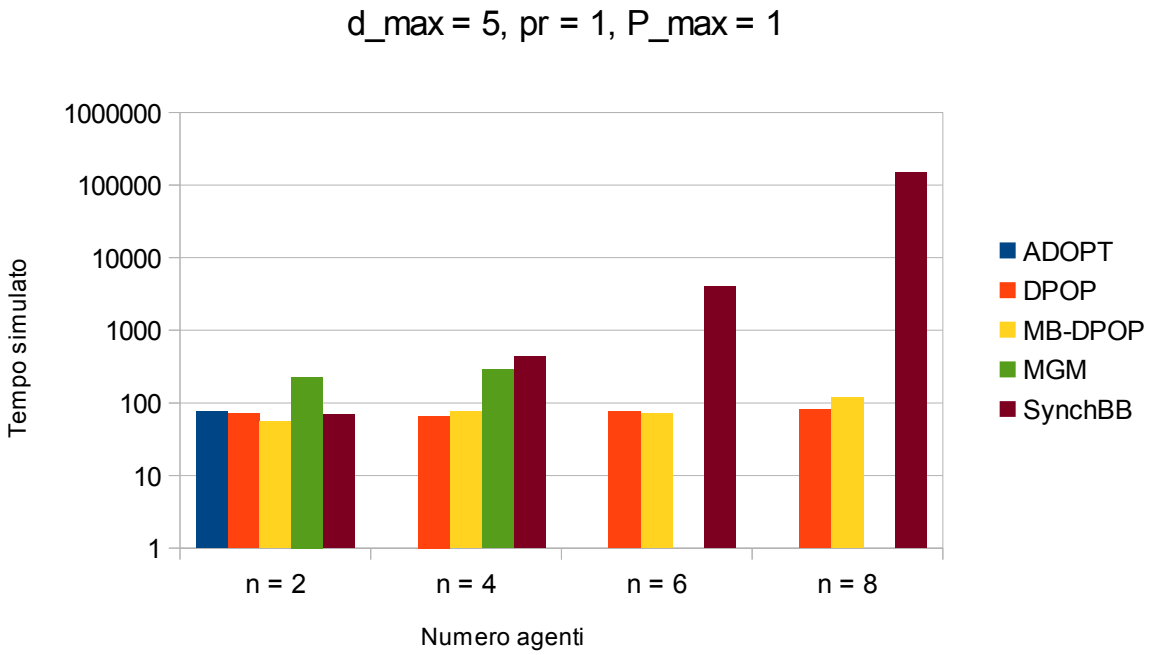

Figura 5.1: Tempo simulato vs. numero di agenti con potenza disponibile 1, una classe di priorità e d_max = 5

La Figura 5.1 mostra come ADOPT sia in grado di funzionare solo con 2 agenti; ciò è dovuto al fatto che questo algoritmo non scala bene all'aumentare della complessità del problema.
MGM non trova mai soluzioni valide con 6 e 8 agenti.
DPOP e MB-DPOP scalano bene all'aumentare del numero di agenti mentre SynchBB dimostra di essere molto meno scalabile.

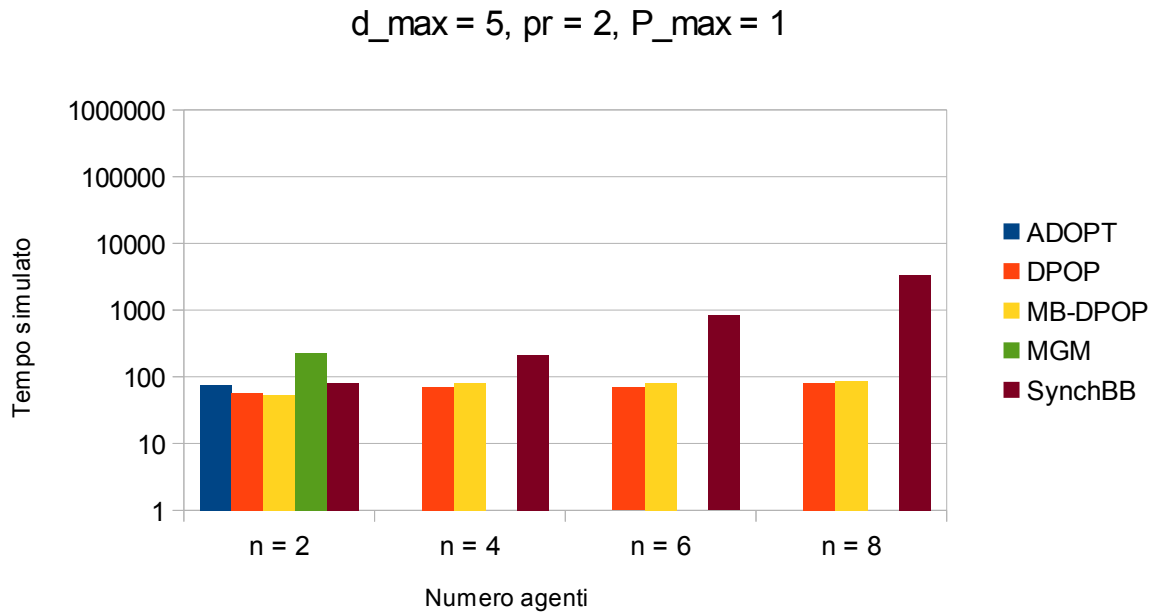

Figura 5.2: Tempo simulato vs. numero di agenti con potenza disponibile 1, due classi di priorità e d_max = 5

La Figura 5.2 illustra che l'aggiunta delle priorità permette a SynchBB di ottenere prestazioni nettamente migliori rispetto al caso precedente. MGM e Adopt funzionano solo con 2 agenti. La famiglia DPOP mantiene le prestazioni inalterate e mostra di non accusare problemi all'aumentare del numero di agenti.

Da questi risultati si può dedurre che la famiglia di algoritmi DPOP è in grado di trovare velocemente la soluzione quando la potenza disponibile è minima; per questo motivo si mostrano ora i risultati dello stesso esperimento con una potenza disponibile diversa.

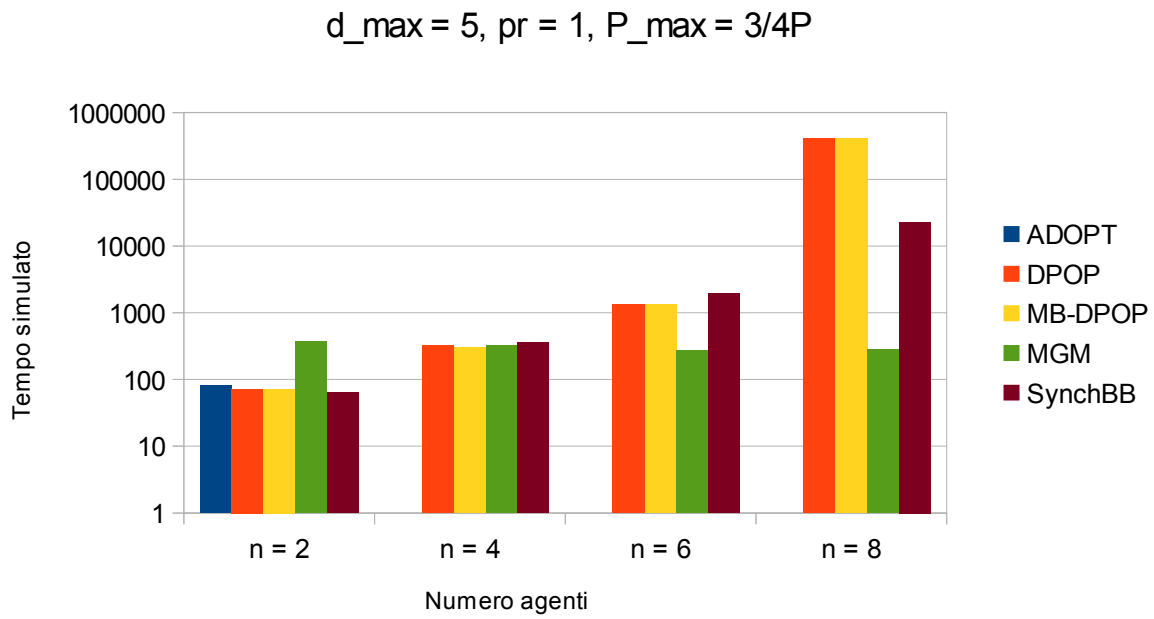

Figura 5.3: Tempo simulato vs. numero di agenti con potenza disponibile 3/4P, una classe di priorità e d_max = 5

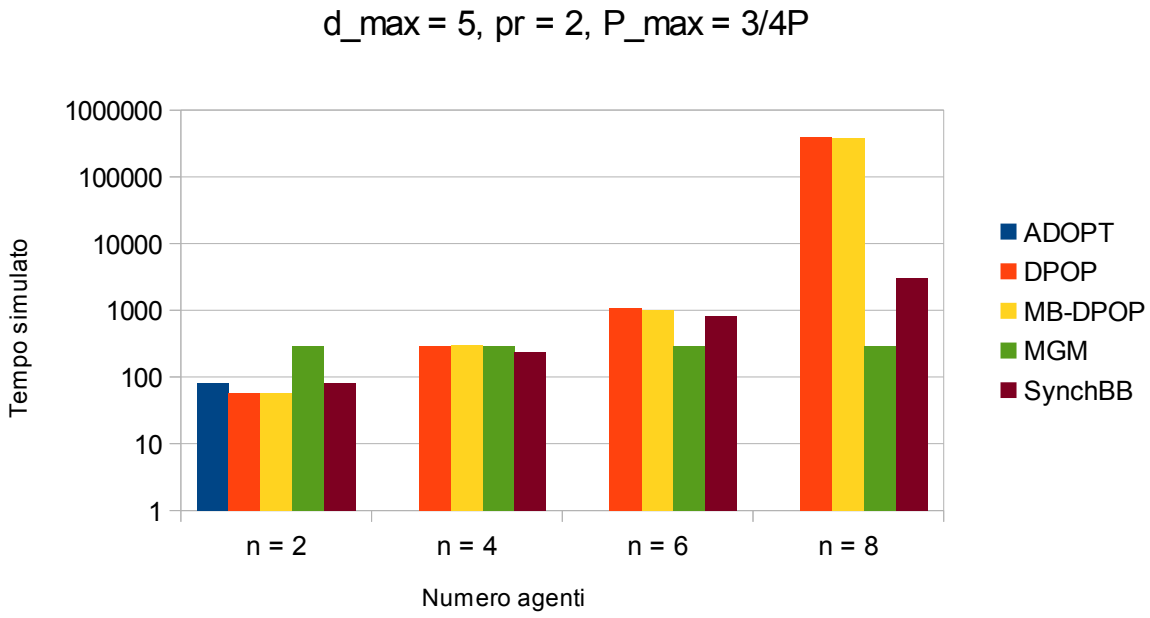

Figura 5.4: Tempo simulato vs. numero di agenti con potenza disponibile 3/4P, due classi di priorità e d_max = 5

Le Figure 5.3 e 5.4 mostrano che che la famiglia DPOP soffre di un notevole peggioramento delle prestazioni all'aumentare del numero di agenti in questa situazione mentre per SynchBB la perdita di prestazioni è più contenuta. Come previsto MGM mantiene un livello di prestazioni stabile in tutti i test e, come in precedenza, SynchBB migliora sensibilmente le proprie prestazioni quando si introducono le classi di priorità.
Si noti che MGM, che nell'esperimento con potenza disponibile pari a 1 trovava raramente la soluzione, ora è sempre in grado di trovare una soluzione valida. Quando la potenza disponibile diminuisce, infatti, lo spazio delle soluzioni rimpicciolisce; ciò è dovuto al fatto che molte tuple devono essere scartate perché causerebbero un utilizzo di potenza superiore alla potenza disponibile. Essendo MGM un algoritmo a ricerca locale è normale osservare che trovare la soluzione diventa, in casi come questo, sempre più difficile; l'algoritmo, infatti, esplora solo una parte dello spazio delle soluzioni e man mano che questo diventa più piccolo è sempre più probabile trovare soluzioni non valide.

Il prossimo esperimento permette di paragonare la qualità delle soluzioni trovate dai vari algoritmi; i problemi usati per l'esperimento sono gli stessi i cui risultati sono illustrati dalle Figure 5.1, 5.2, 5.3 e 5.4. Si ricorda che un valore inferiore è migliore perché corrisponde a un costo della soluzione minore.

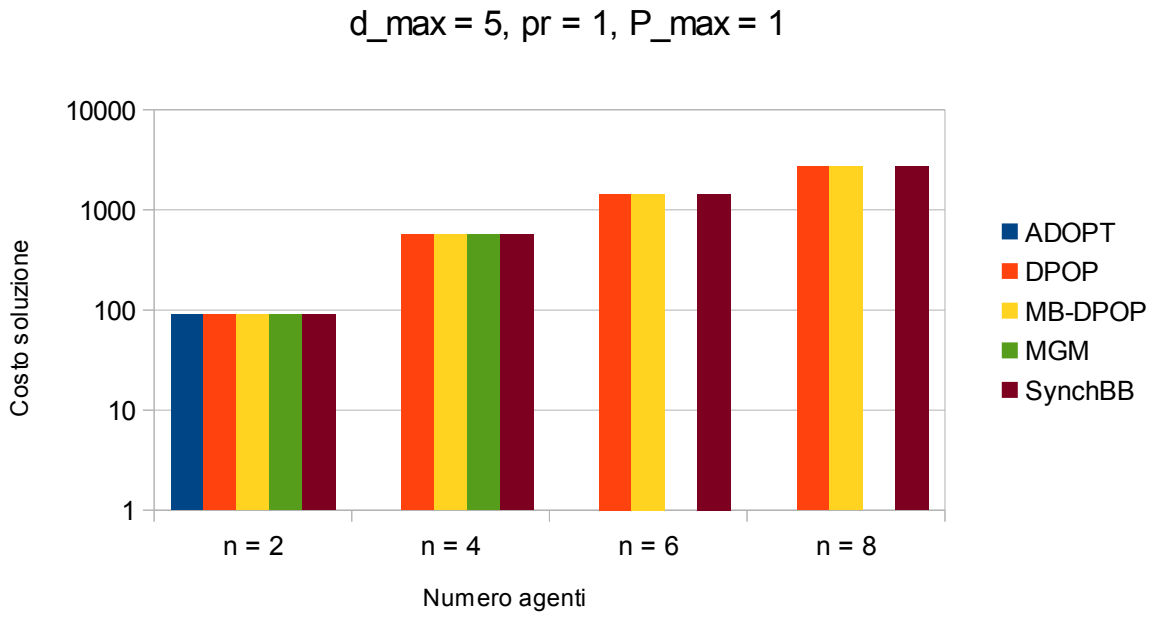

Figura 5.5: Costo soluzione vs. numero di agenti con potenza disponibile 1, una classe di priorità e d_max = 5

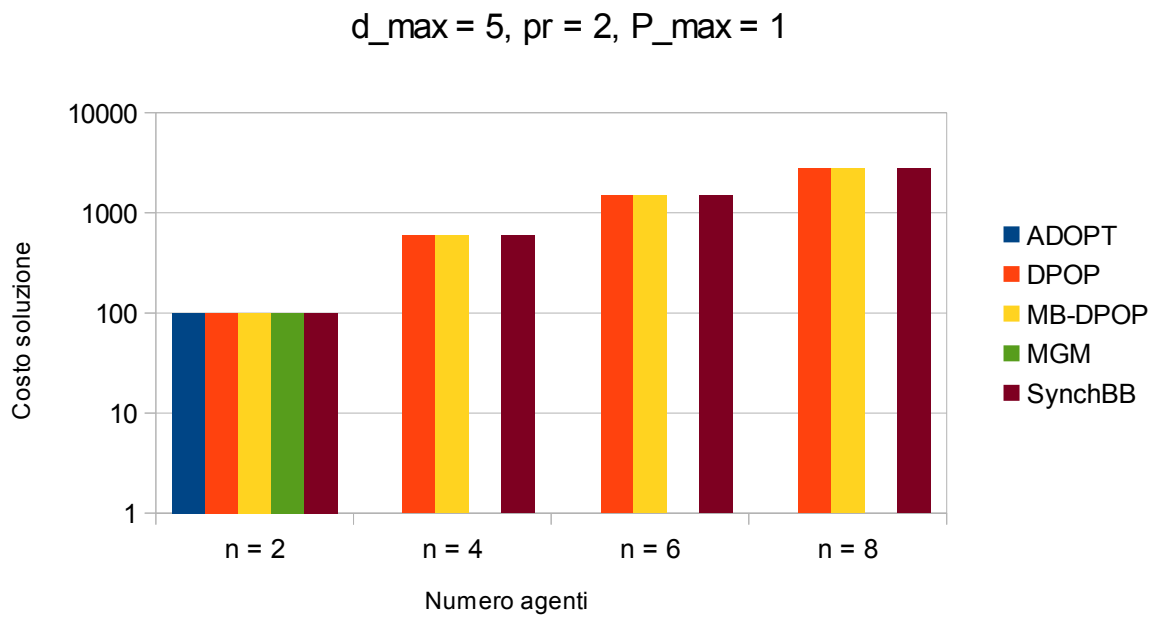

Figura 5.6: Costo soluzione vs. numero di agenti con potenza disponibile 1, due classi di priorità e d_max = 5

Le Figure 5.5 e 5.6 mostrano che con potenza disponibile pari a 1 non ci sono differenze nella qualità della soluzione; l'unico problema è che spesso MGM non è in grado di trovare una soluzione valida.

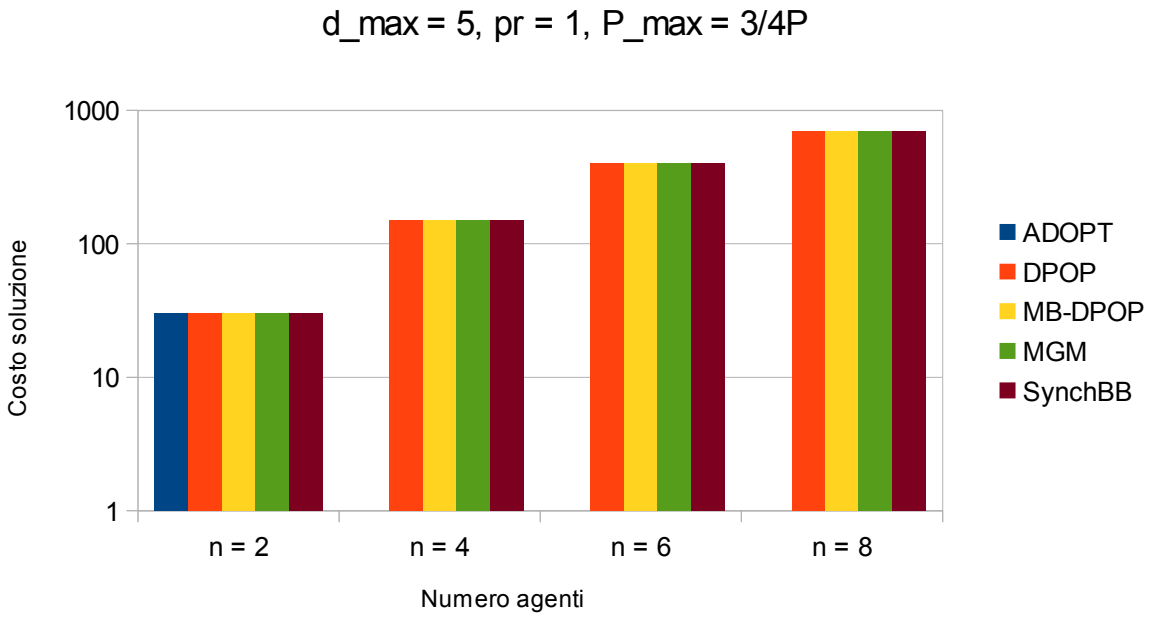

Figura 5.7: Costo soluzione vs. numero di agenti con potenza disponibile 3/4P, una classe di priorità e d_max = 5

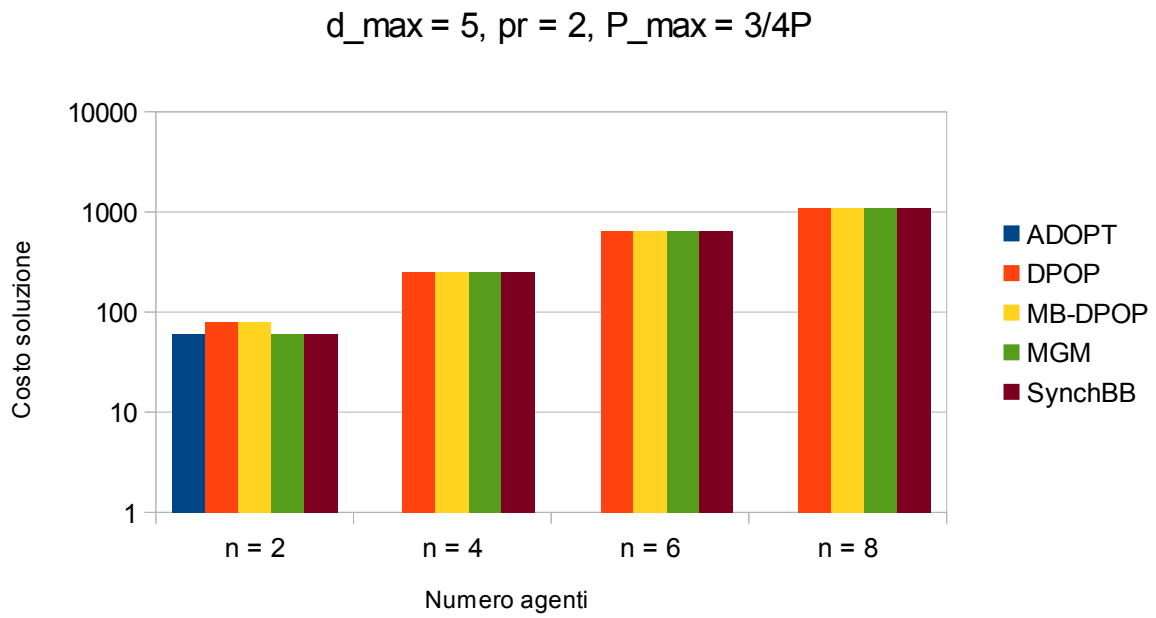

Figura 5.8: Costo soluzione vs. numero di agenti con potenza disponibile 3/4P, due classi di priorità e d_max = 5

Le Figure 5.7 e 5.8 mostrano quando la potenza disponibile è pari a 3/4P in un caso la famiglia di algoritmi DPOP ha trovato una soluzione non ottima; in tutti gli altri casi le soluzioni trovate sono tutte ottime.

5.3 Analisi delle prestazioni in base alla cardinalità del dominio

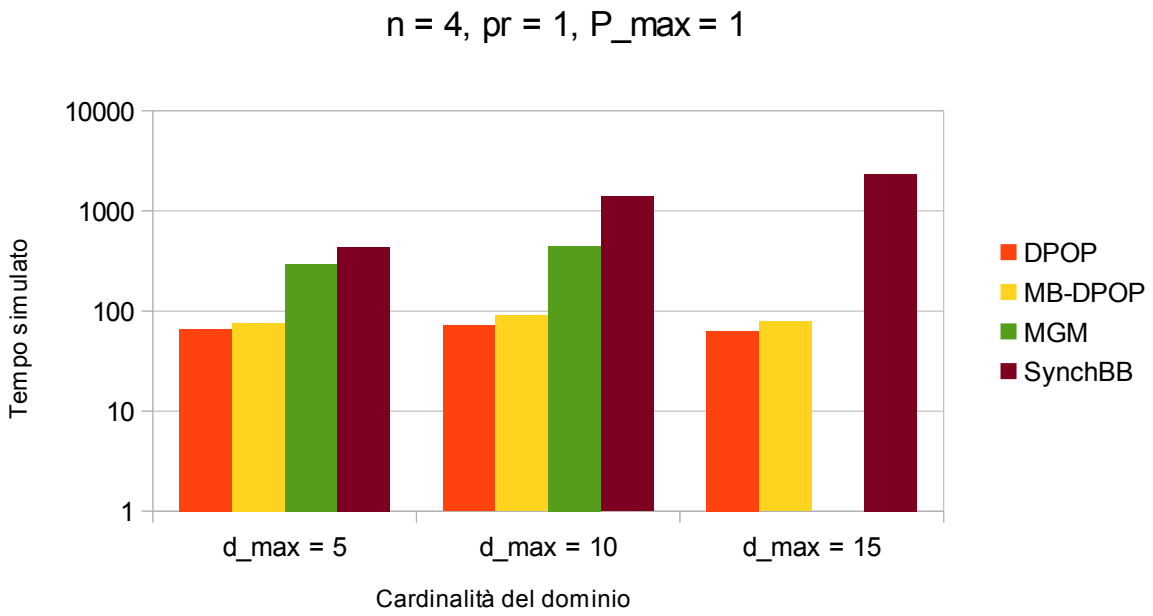

Figura 5.9: Tempo simulato vs. cardinalità del dominio con potenza disponibile 1, una classe di priorità e 4 agenti

Nella Figura 5.9 si nota che SynchBB mostra una marcata perdita di prestazioni all'aumentare della cardinalità del dominio mentre la famiglia DPOP mantiene tempi di esecuzione costanti. MGM non presenta problemi prestazionali ma non riesce a trovare la soluzione quando d_max è pari a 15, probabilmente a causa dell'aumento della dimensione dello spazio delle soluzioni.

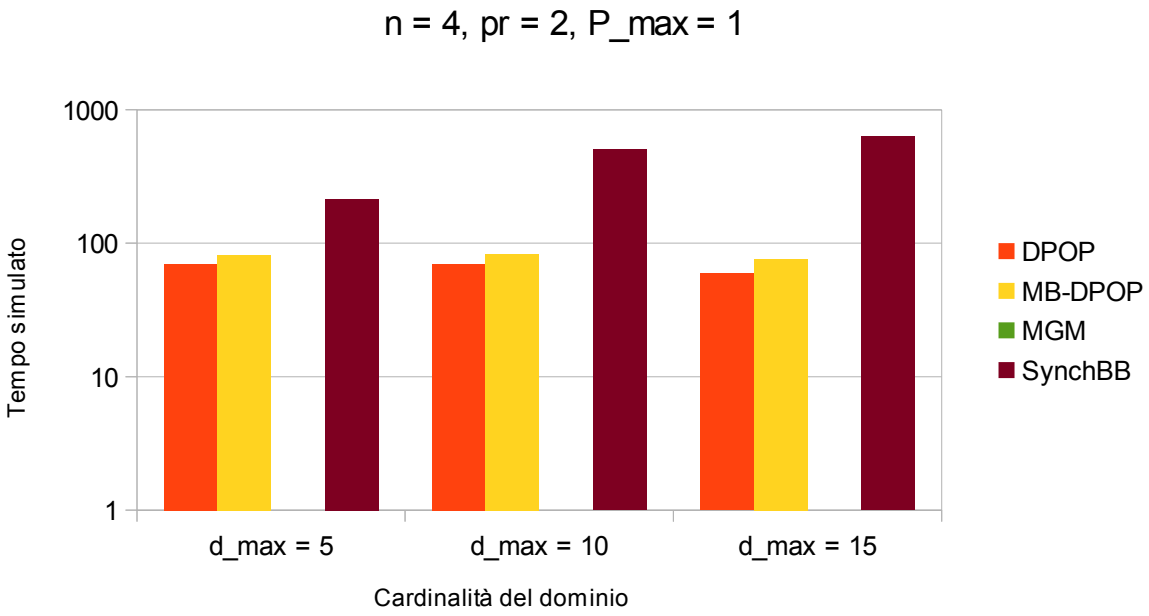

Figura 5.10: Tempo simulato vs. cardinalità del dominio con potenza disponibile 1, due classi di priorità e 4 agenti

La Figura 5.10 illustra che l'introduzione delle priorità aiuta molto SynchBB a raggiungere più velocemente la soluzione; al contrario MGM non riesce mai a trovare la soluzione, probabilmente a causa delle condizioni aggiuntive imposte dalle classi di priorità.

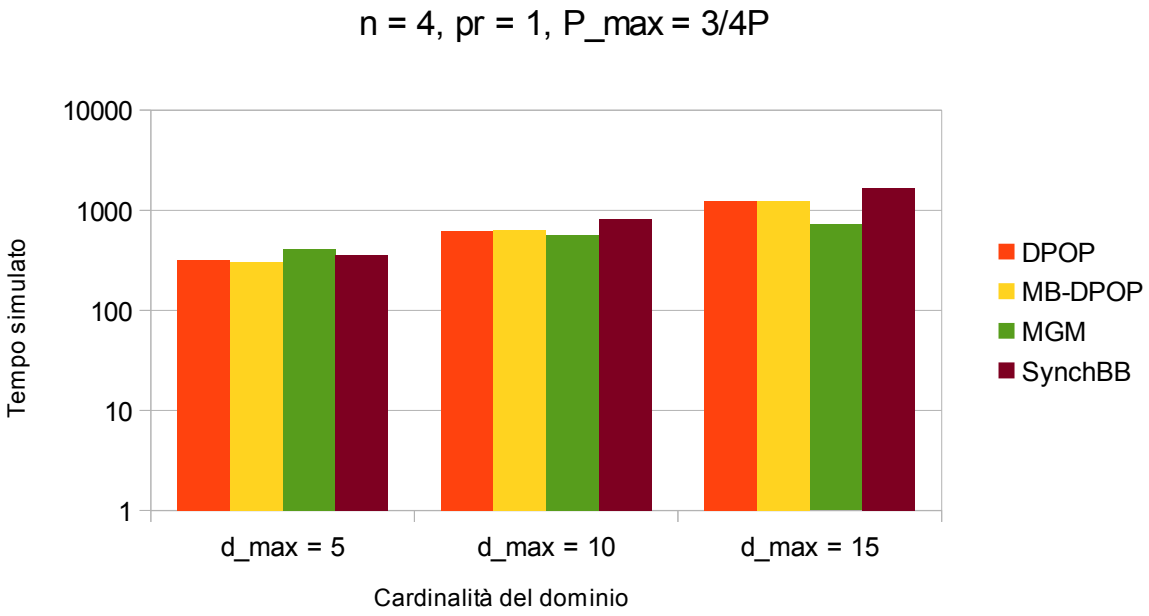

Figura 5.11: Tempo simulato vs. cardinalità del dominio con potenza disponibile 3/4P, una classe di priorità e 4 agenti

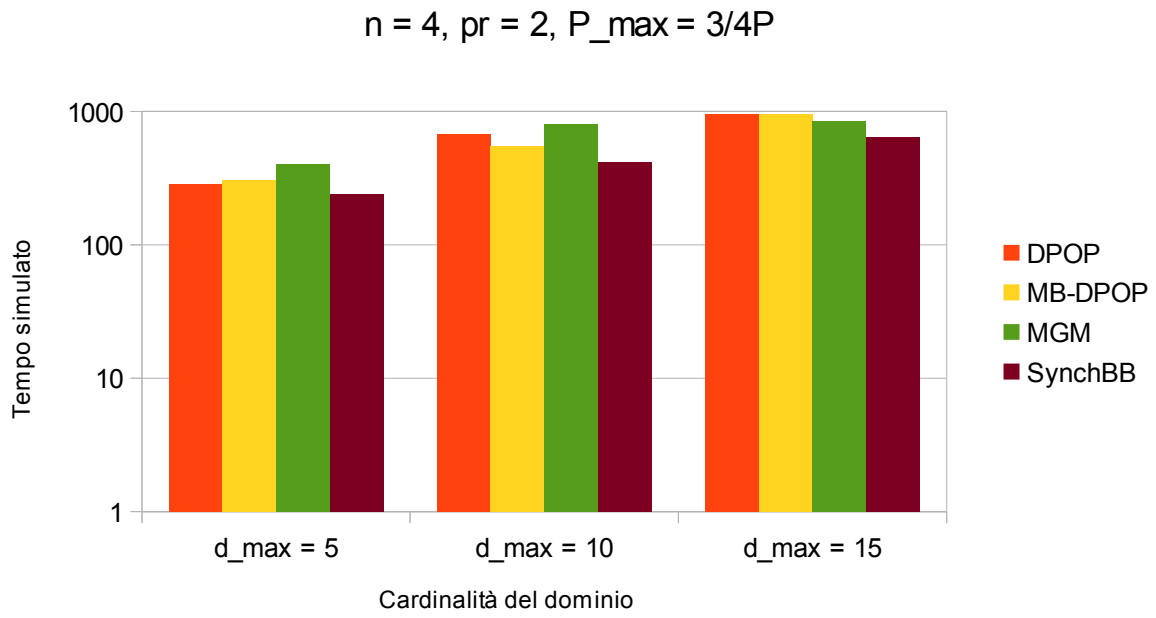

Figura 5.12: Tempo simulato vs. cardinalità del dominio con potenza disponibile 3/4P, due classi di priorità e 4 agenti

Le Figure 5.11 e 5.12 dimostrano che nel momento in cui la soluzione non è più unica MGM riesce a trovare la soluzione, anche se le sue prestazioni sono in alcuni casi superiori a quelle degli algoritmi a ricerca globale. Con una sola classe di priorità SynchBB ha tempi di esecuzione che si dilatano notevolmente all'aumentare della cardinalità del dominio. L'introduzione delle priorità ribalta la soluzione e permette a SynchBB di ottenere prestazioni sempre migliori rispetto alla famiglia DPOP.

5.4 Analisi delle prestazioni in base alle classi di priorità

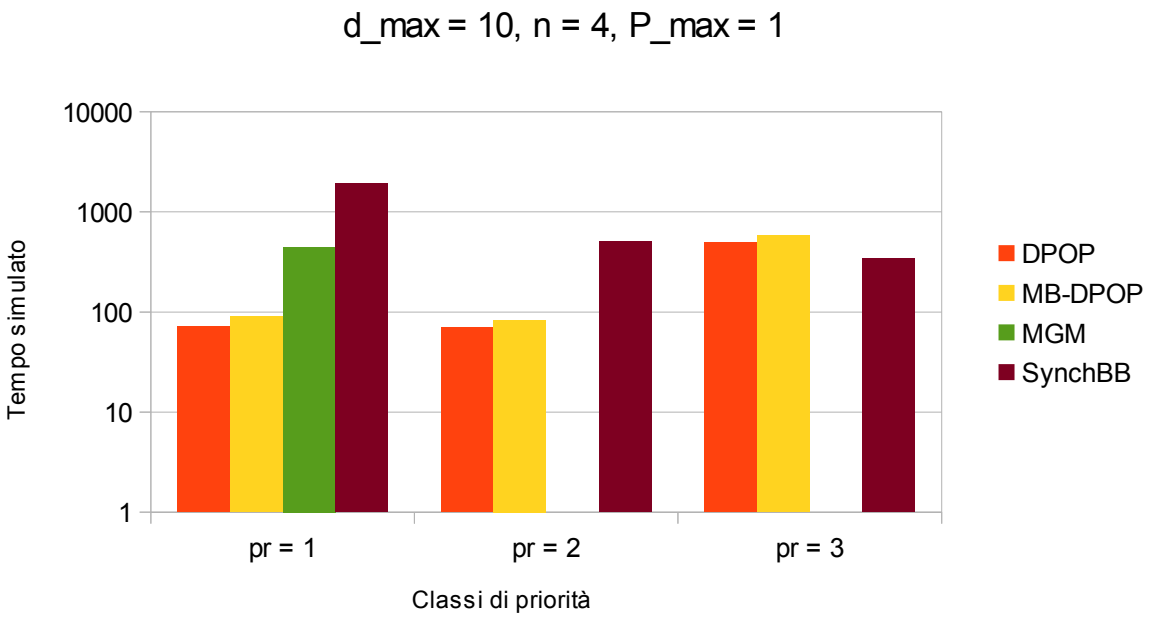

Figura 5.13: Tempo simulato vs. classi di priorità con potenza disponibile 1, 4 agenti e d_max = 10

La Figura 5.13 mostra che, come già fatto notare precedentemente, i problemi con potenza disponibile pari a 1 causano problemi a MGM che non è in grado di fornire una soluzione nel momento in cui si introducono le priorità. La famiglia DPOP subisce un peggioramento delle prestazioni all'aumentare delle priorità mentre SynchBB mostra il comportamento contrario e beneficia molto dei vincoli più stringenti introdotti dalle priorità.

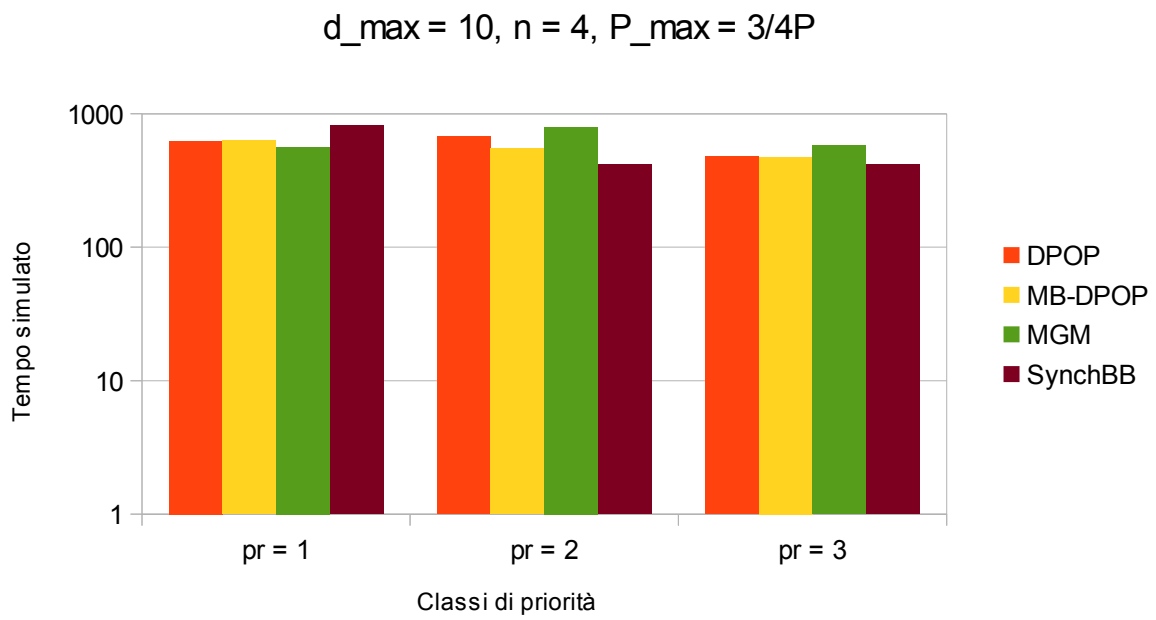

Figura 5.14: Tempo simulato vs. classi di priorità con potenza disponibile 3/4P, 4 agenti e d_max = 10

La Figura 5.14 mostra che aumentando la potenza a 3/4P i comportamenti degli algoritmi cambiano. Le prestazioni si avvicinano molto e tutti gli algoritmi tranne SynchBB mostrano di non subire gli effetti delle priorità. SynchBB continua ad ottenere benefici dalle priorità tanto da diventare l'algoritmo dalle prestazioni migliori nel momento in cui le priorità vengono introdotte.

5.5 Analisi delle prestazioni in base alla potenza disponibile

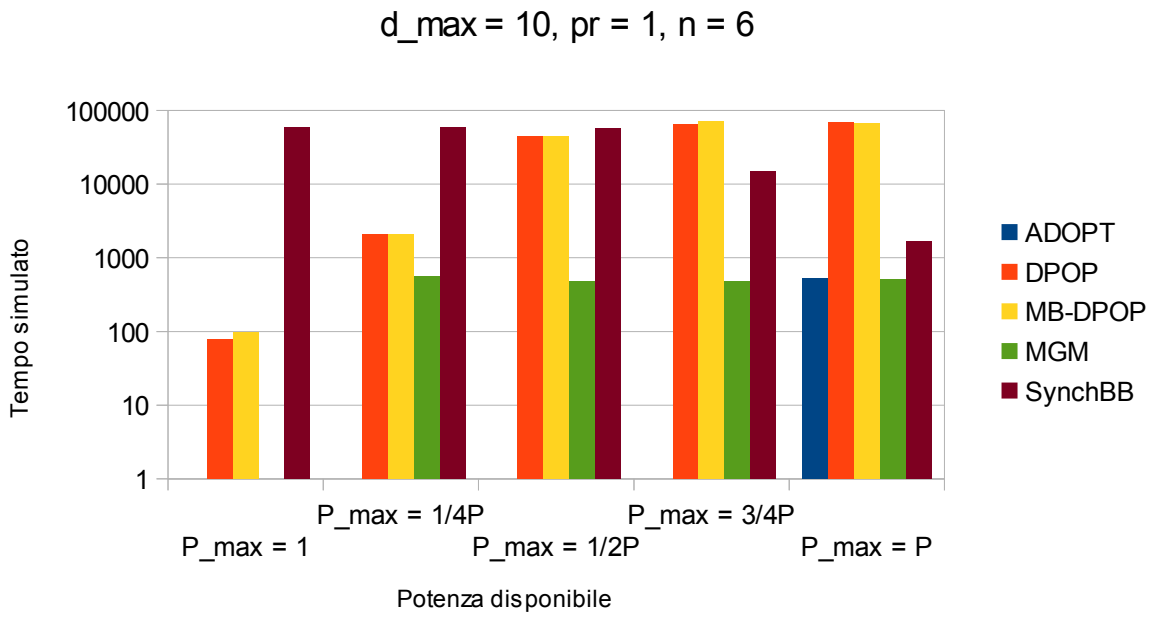

Figura 5.15: Tempo simulato vs. potenza disponibile con 6 agenti, d_max = 10 e una classe di priorità

La Figura 5.15 dimostra che gli algoritmi si comportano in maniera diversa a seconda della potenza disponibile: ADOPT funziona solo se la potenza disponibile è sufficiente a coprire completamente il fabbisogno dei carichi. MGM, come spesso accade, non funziona se la potenza disponibile è 1 e in tutti gli altri casi ha prestazioni costanti. La famiglia DPOP vede le proprie prestazioni peggiorare significativamente all'aumentare della potenza mentre l'esatto contrario accade con SynchBB.

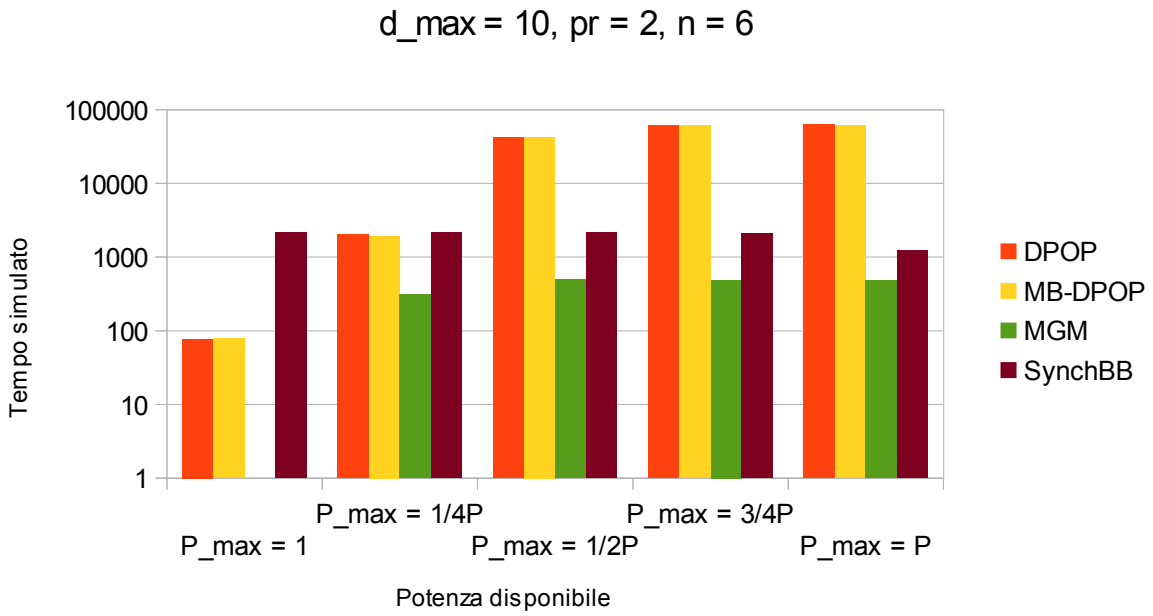

Figura 5.16: Tempo simulato vs. potenza disponibile con 6 agenti, d_max = 10 e due classi di priorità

La Figura 5.16 mostra che con due classi di priorità SynchBB ottiene quasi sempre prestazioni migliori o paragonabili a quelle della famiglia DPOP; MGM, tranne il solito caso particolare, ottiene sempre prestazioni di gran lunga migliori rispetto a tutti gli altri algoritmi.

5.6 Analisi dei messaggi scambiati in base alla cardinalità del dominio

I precedenti esperimenti mostrano come, al variare di parametri quali cardinalità del dominio, classi di priorità e numero di agenti, le prestazioni degli algoritmi cambiano. Si vuole ora mostrare come variano i messaggi scambiati in modo da poter tracciare un parallelo tra tempo simulato e messaggi scambiati.

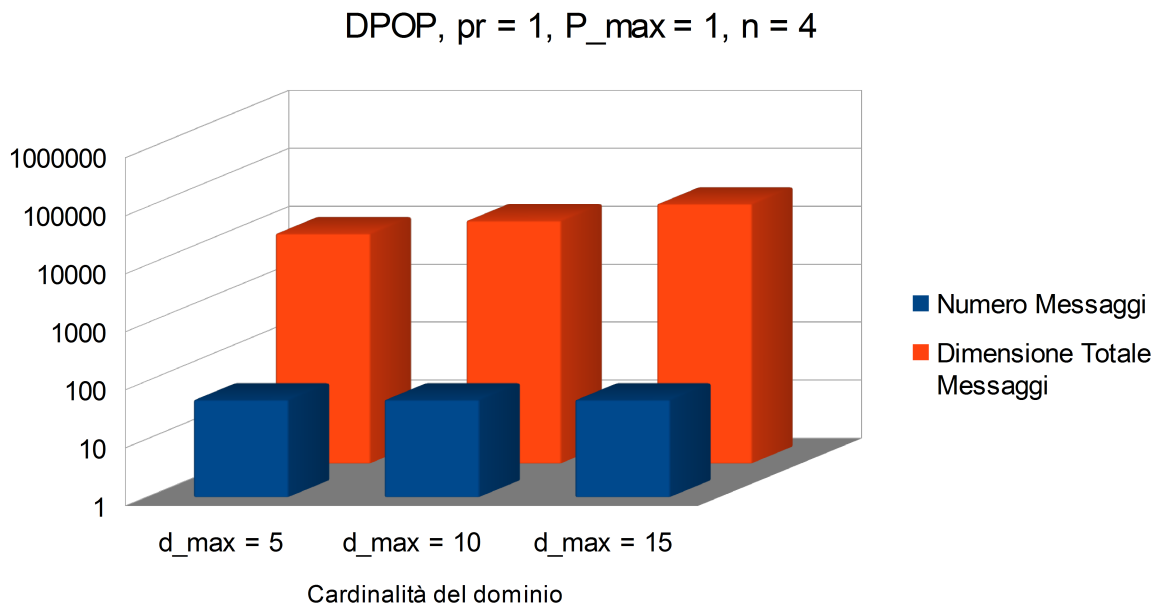

Figura 5.17: DPOP - Messaggi scambiati vs. cardinalità del dominio con potenza disponibile 1, 4 agenti e una classe di priorità

In Figura 5.17 si nota che all'aumentare della cardinalità del dominio DPOP scambia lo stesso numero di messaggi ma la loro dimensione cresce.

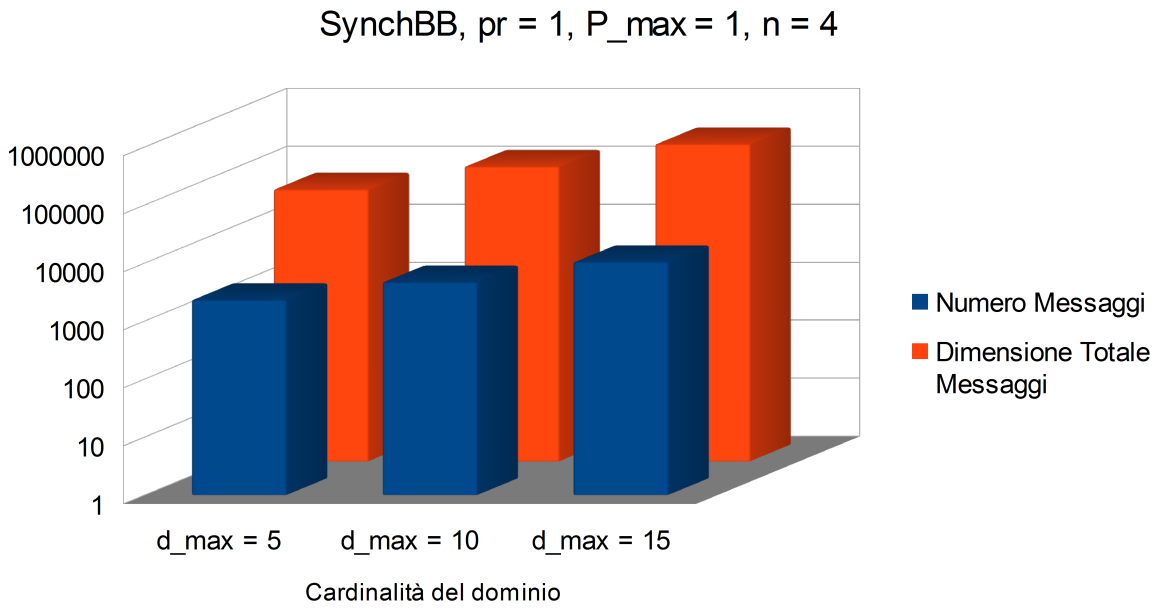

Figura 5.18: SynchBB - Messaggi scambiati vs. cardinalità del dominio con potenza disponibile 1, 4 agenti e una classe di priorità

La Figura 5.18 mostra che SynchBB scambia molti più messaggi all'aumentare della cardinalità del dominio e anche la loro dimensione aumenta. Paragonato a DPOP la quantità di informazioni scambiate è nettamente superiore. I risultati di questo esperimento dimostrano una correlazione tra messaggi scambiati e tempo simulato.

5.7 Analisi dei messaggi scambiati in base alle classi di priorità

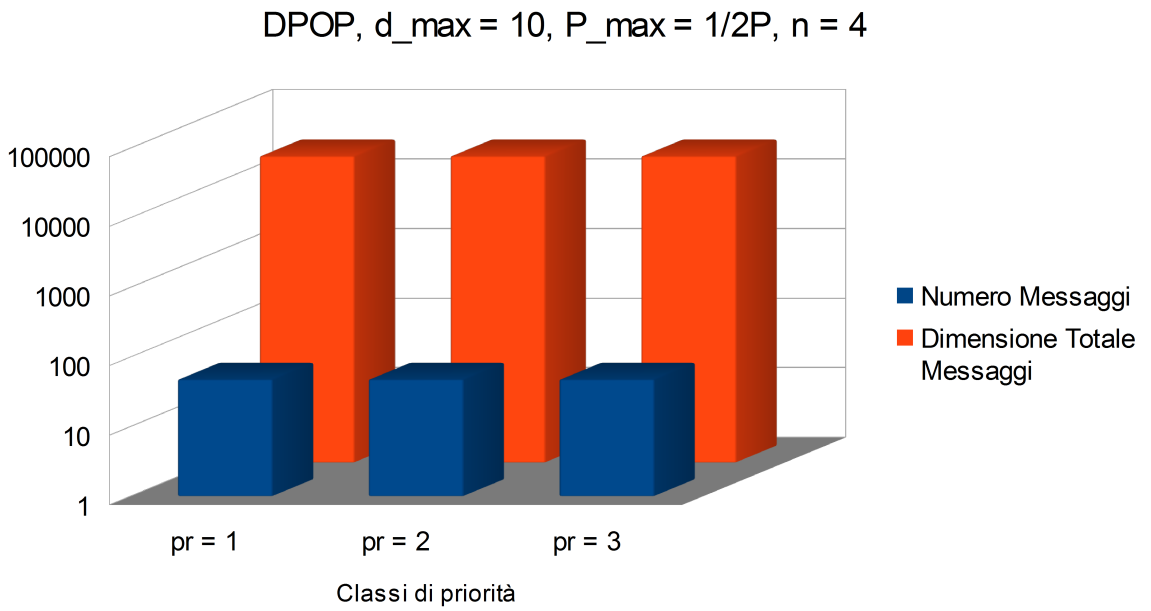

Figura 5.19: DPOP - Messaggi scambiati vs. classi di priorità con potenza disponibile 1/2P, 4 agenti e d_max = 10

In Figura 5.19 si nota come DPOP non subisce alcuna variazione dovuta all'introduzione delle priorità.

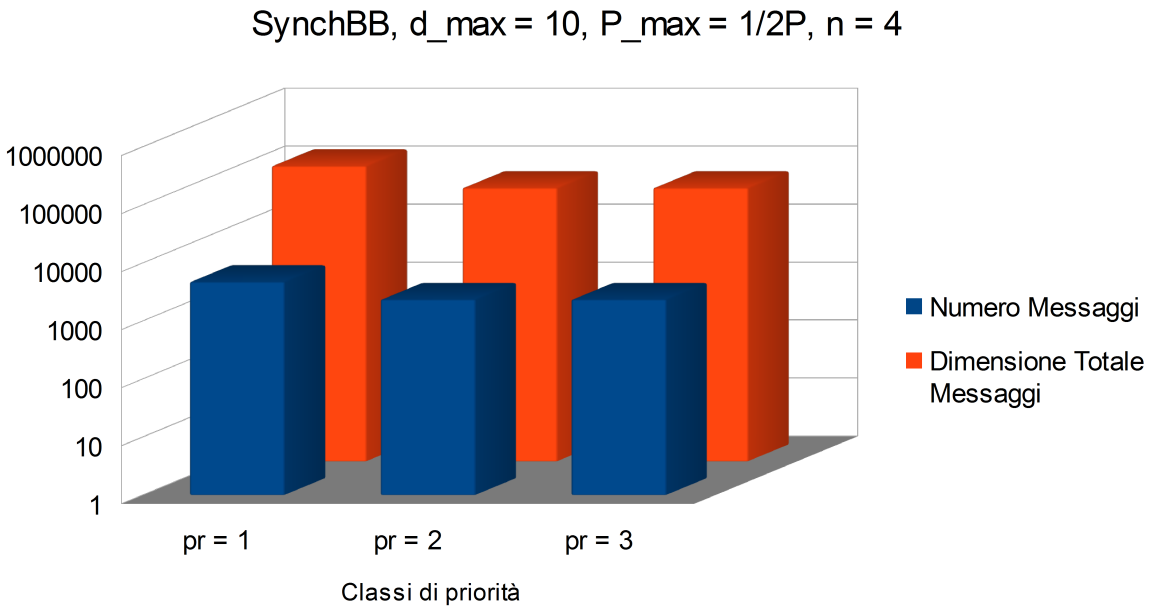

Figura 5.20: SynchBB - Messaggi scambiati vs. classi di priorità con potenza disponibile 1/2P, 4 agenti e d_max = 10

La Figura 5.20 dimostra che SynchBB è sensibile alle priorità: il passaggio da una a due priorità genera una distinta diminuzione dei messaggi; nessun cambiamento nel passaggio da 2 a 3 priorità. Nelle prove precedenti si è sempre notato che SynchBB migliora le proprie prestazioni con l'introduzione delle priorità; questo esperimento fornisce una giustificazione a tale comportamento.

5.8 Analisi dei messaggi scambiati in base alla potenza disponibile

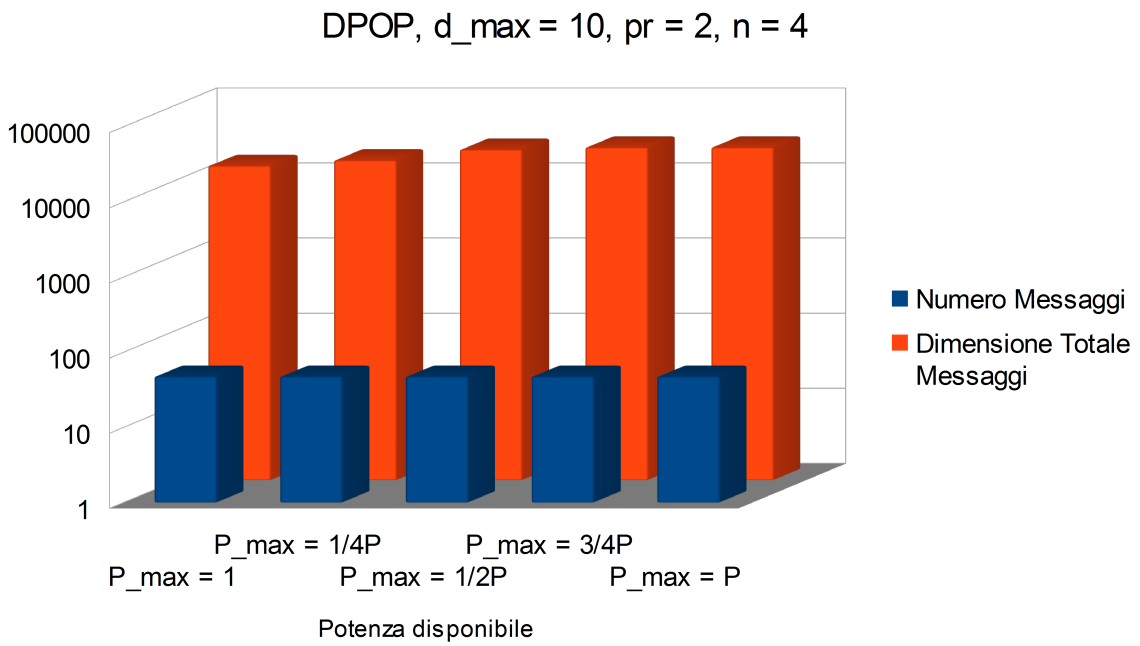

Figura 5.21: DPOP - Messaggi scambiati vs. potenza disponibile con 4 agenti, due classi di priorità e d_max = 10

La Figura 5.21 mostra che con DPOP il numero di messaggi scambiati rimane costante all'aumentare della potenza disponibile, mentre la loro dimensione cresce leggermente.

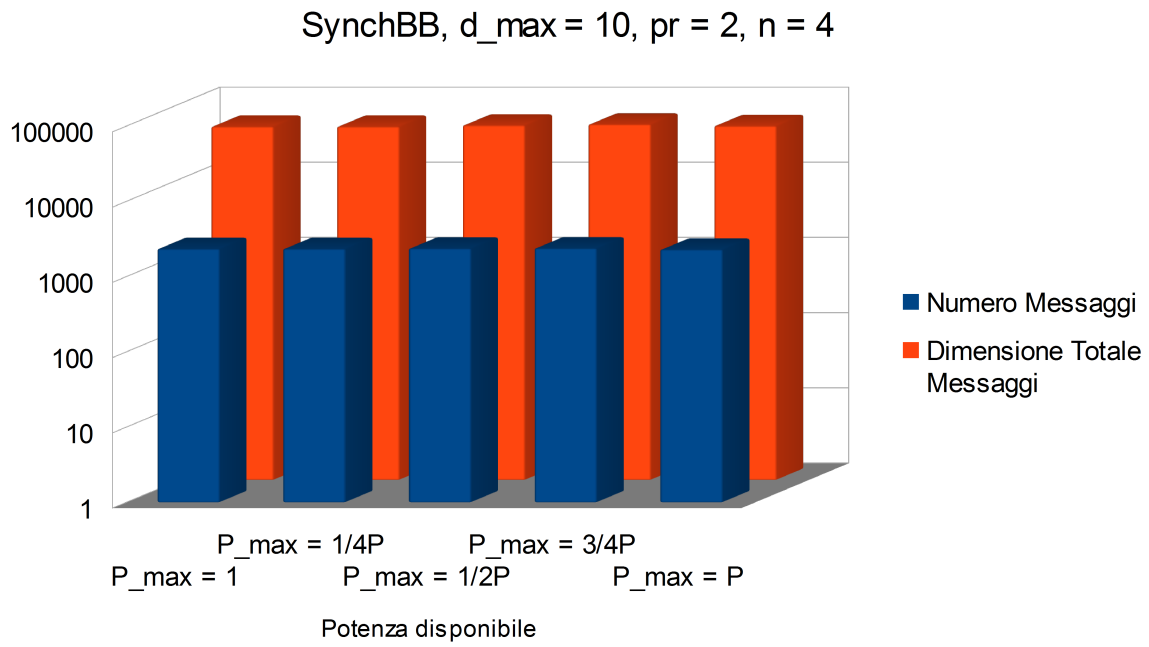

Figura 5.22: SynchBB - Messaggi scambiati vs. potenza disponibile con 4 agenti, due classi di priorità e d_max = 10

In Figura 5.22 si nota come SynchBB offra risultati difficilmente decifrabili: le variazioni del numero di messaggi scambiati e della loro dimensione sono minime e non sembrano dipendere dall'incremento della potenza disponibile. In questo caso non è quindi possibile stabilire una correlazione tra tempo simulato e messaggi scambiati.

Seguono due grafici che mettono a confronto i messaggi scambiati da più algoritmi nel risolvere lo stesso problema.

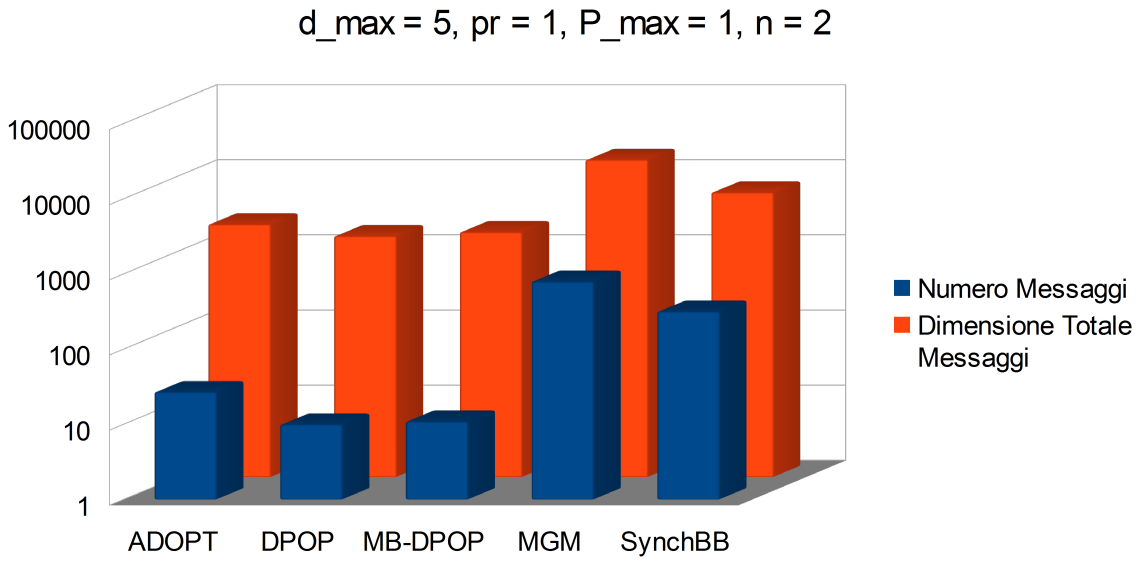

Figura 5.23: Messaggi scambiati con 2 agenti, una classe di priorità, potenza disponibile 1 e d_max = 5

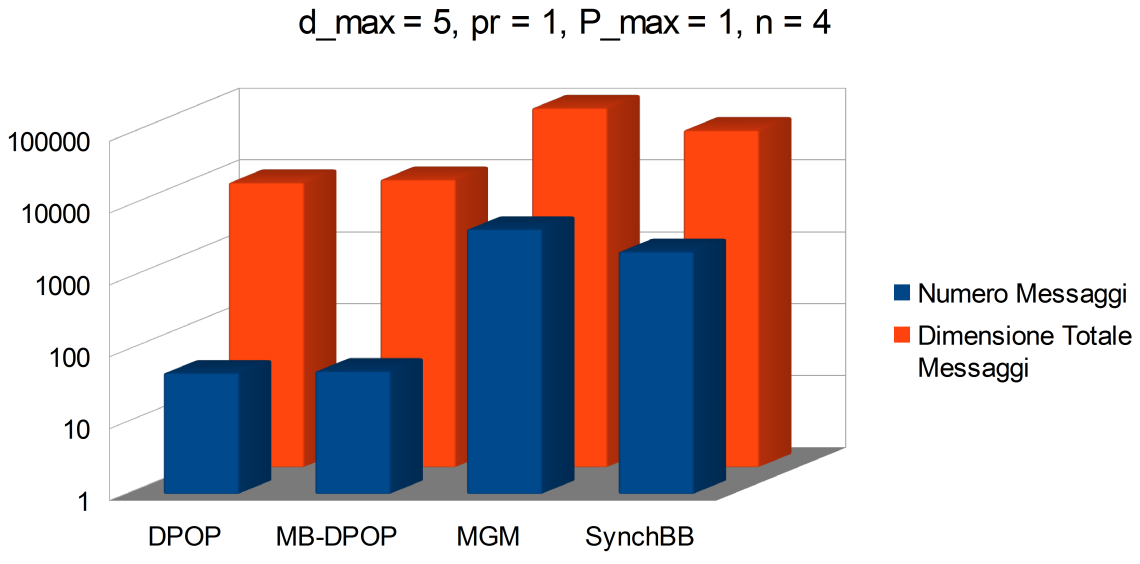

Figura 5.24: Messaggi scambiati con 4 agenti, una classe di priorità, potenza disponibile 1 e d_max = 5

Le Figure 5.23 e 5.24 mostrano chiaramente che MGM usa di gran lunga più messaggi degli altri algoritmi e che anche la dimensione dei messaggi scambiati è molto superiore; SynchBB

ha prestazioni migliori di MGM mentre la famiglia DPOP scambia pochi messaggi.
L'aumento del numero degli agenti causa un aumento dei messaggi scambiati piuttosto consistente in tutti gli algoritmi; questo spiega il motivo per il quale FRODO non è in grado di fornire informazioni sui messaggi quando il numero di agenti supera 4.

Nell'esaminare i risultati si è notato che l'introduzione delle classi di priorità non genera un aumento del numero di messaggi scambiati o della loro dimensione. Anche l'incremento della cardinalità del dominio non genera un aumento dei messaggi scambiati, anche se nel caso di MGM la dimensione dei messaggi è aumentata notevolmente.

5.9 Analisi delle prestazioni dell'algoritmo MGM

Come già spiegato MGM è un algoritmo a ricerca locale, il che implica che potrebbe non trovare una soluzione anche quando questa esiste. D'altra parte gli algoritmi a ricerca locale hanno spesso prestazioni migliori rispetto agli algoritmi a ricerca globale e scalano meglio all'aumentare del numero di agenti e della cardinalità del dominio, come dimostrato dagli esperimenti precedentemente illustrati.

Si ritiene, dunque, che sia utile e interessante analizzare la qualità delle soluzioni trovate da MGM al fine di meglio valutare le caratteristiche di questo algoritmo.

In questa sezione verranno analizzate le soluzioni trovate da MGM in base a due metriche: la capacità di trovare una qualunque soluzione valida a un problema e la capacità di trovare la soluzione ottima.

Al fine di valutare le prestazioni ottenute tutti i problemi descritti nella Sezione 5.1 sono stati eseguiti 10 volte con MGM. I seguenti grafici mostrano con che percentuale MGM ha raggiunto gli obiettivi (trovare una soluzione e trovare la soluzione ottima). I seguenti grafici, a differenza di tutti gli altri presenti nel capitolo, hanno un asse delle ordinate che usa una scala lineare e non logaritmica.

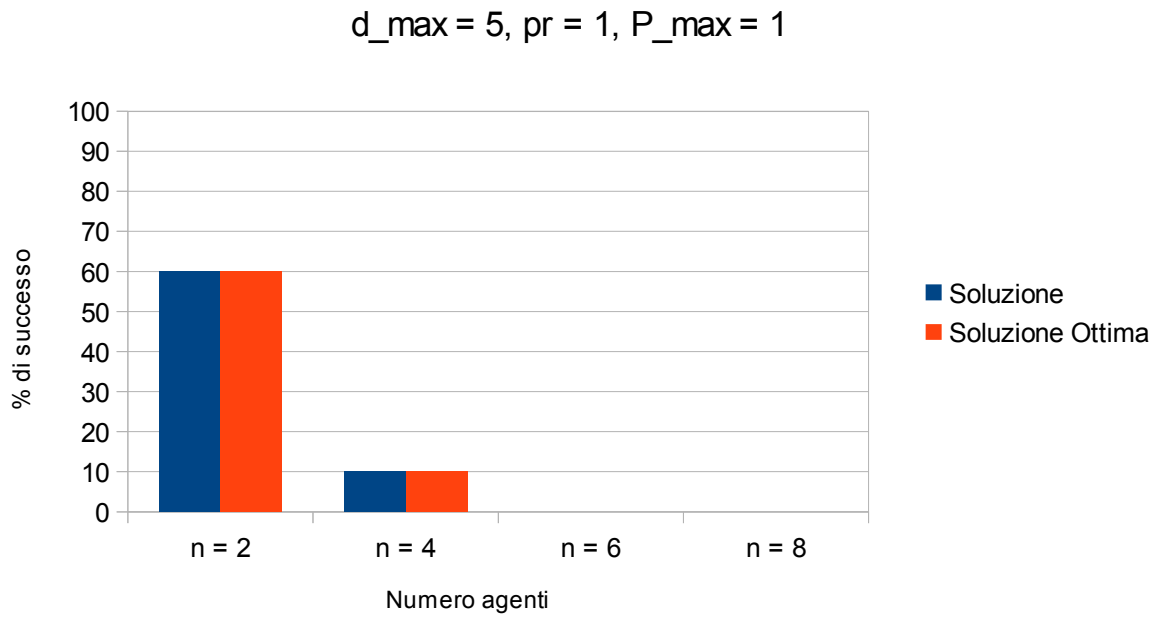

Figura 5.25: Qualità della soluzione vs. numero di agenti con potenza massima 1, una classe di priorità e d_max = 5

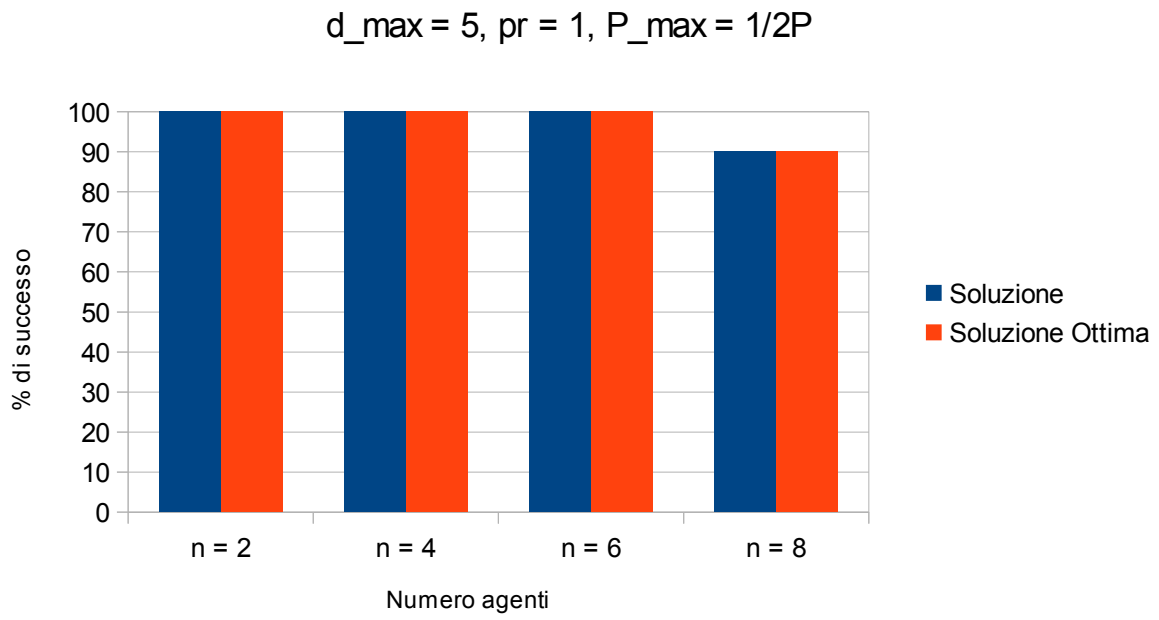

Figura 5.26: Qualità della soluzione vs. numero di agenti con potenza massima 1/2P, una classe di priorità e d_max = 5

La Figura 5.25 mostra che all'aumentare degli agenti la capacità di MGM di trovare una soluzione crolla; d'altra parte è noto dai risultati degli esperimenti precedenti che quando la potenza disponibile è pari a 1 MGM fa fatica a funzionare. Per questo motivo l'esperimento viene ora ripetuto con P_max = 1/2P.
In Figura 5.26 si nota come basti aumentare la potenza disponibile e MGM ottiene immediatamente prestazioni di gran lunga superiori; solo con 8 agenti si registra un caso in cui l'algoritmo non riesce a trovare la soluzione.

Visti i risultati davvero mutevoli si è deciso di valutare come le prestazioni di MGM variano in base anche ad altri parametri.

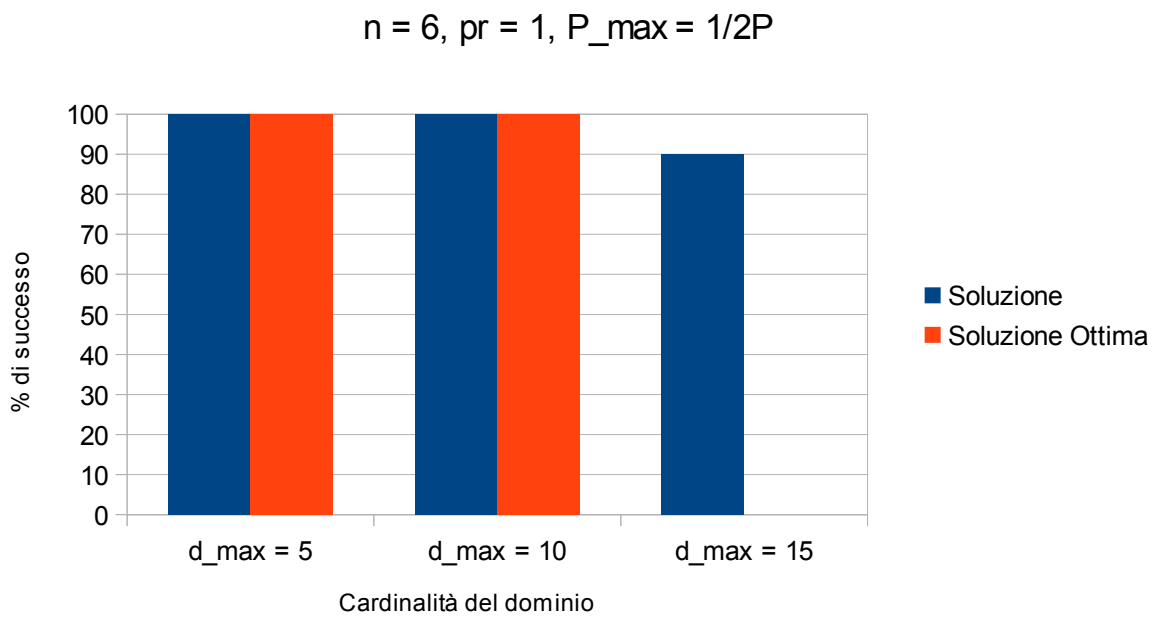

Figura 5.27: Qualità della soluzione vs. cardinalità del dominio con 6 agenti, potenza massima 1/2P e una classe di priorità

La Figura 5.27 mostra che con una cardinalità elevata MGM riesce, in generale, a trovare una soluzione ma la capacità di trovare la soluzione ottima diminuisce bruscamente. Lo stesso esperimento viene ora eseguito con differenti classi di priorità.

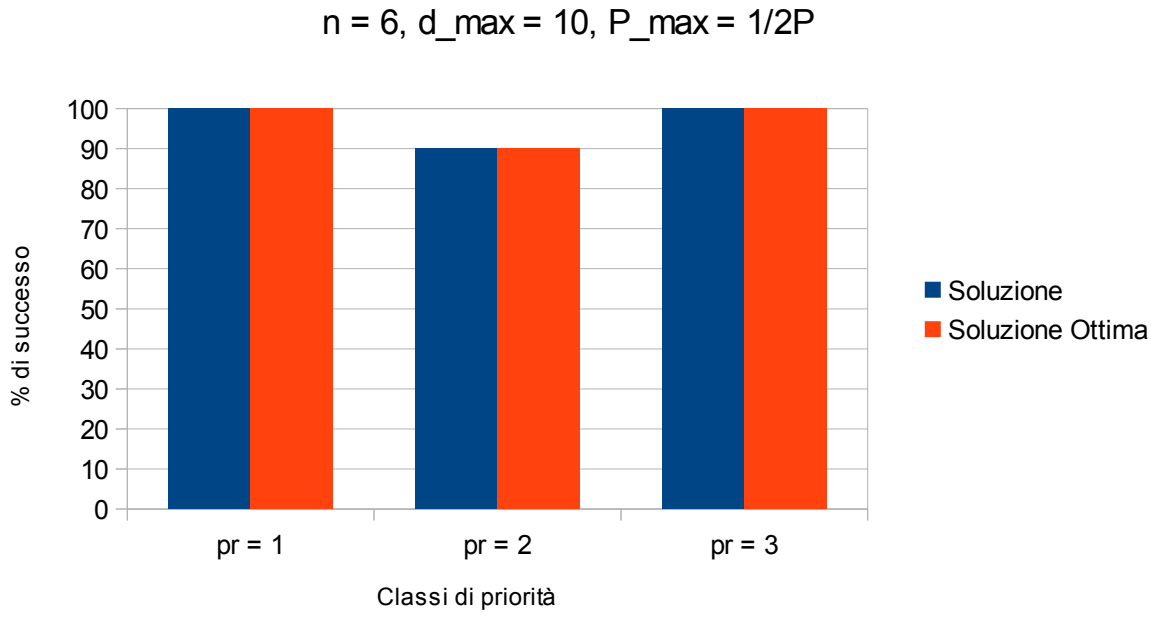

Figura 5.28: Qualità della soluzione vs. classi di priorità con 6 agenti, potenza massima 1/2P e d_max = 10

La Figura 5.28 mostra che le classi di priorità non sembrano avere un impatto sulle prestazioni di MGM. Si valutano ora le prestazioni di MGM al variare della potenza disponibile.

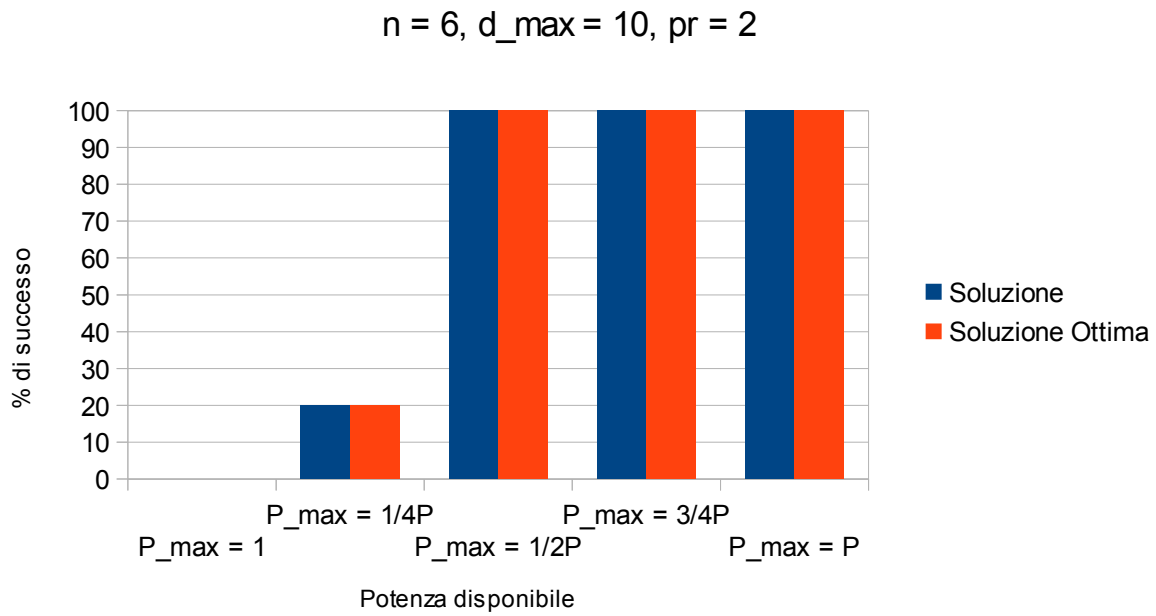

Figura 5.29: Qualità della soluzione vs. potenza disponibile con 6 agenti, due classi di priorità e d_max = 10

La Figura 5.29 mostra che la potenza disponibile ha un grande impatto sulla soluzione trovata da MGM; come già spiegato una potenza bassa rende più piccolo lo spazio delle soluzioni e ciò ha un notevole impatto su un algoritmo a ricerca locale come MGM.

Capitolo 6

Questo capitolo porta a compimento il presente lavoro di tesi con alcune considerazioni riguardanti i risultati ottenuti e le possibili espansioni di questo lavoro.
La Sezione 6.1 contiene alcune riflessioni sui risultati ottenuti.
La Sezione 6.2 offre alcuni spunti su eventuali sviluppi di questo lavoro e su ulteriori esperimenti che andrebbero eseguiti per meglio valutare il modello e la sua applicabilità a reti elettriche reali.

6.1 Considerazioni sui risultati ottenuti

Gli esperimenti descritti nel capitolo precedente hanno dimostrato che il modello analizzato funziona correttamente. L'energia disponibile, infatti, viene distribuita agli agenti rispettando la priorità assegnata ad ognuno; inoltre il modello evita gli sprechi di energia aumentando il costo della soluzione in caso di inefficienze nella distribuzione dell'energia.
Le prestazioni degli algoritmi impongono, però, una riflessione sul tipo di rete elettrica che può essere gestita tramite il sistema multiagente preso in considerazione in questa tesi. Si è notata, infatti, una estrema variabilità dei risultati che rende complicato decidere quale algoritmo sia più adatto a trovare la soluzione al problema.
Ad esempio si è notato che ogni algoritmo è sensibile ai parametri utilizzati per creare il problema: MGM fa fatica a trovare la soluzione quando la potenza disponibile è 1, SynchBB è lento quando non ci sono agenti che lavorano a priorità diverse, DPOP a volte ha prestazioni nettamente superiori a quelle di SynchBB.
Si ritiene dunque che una persona eventualmente interessata a gestire una rete sfruttando il modello valutato dovrebbe porsi le seguenti domande:

1. Quanti agenti, in media, sono presenti nella rete?
2. Che dominio ha ogni agente?
3. È utile (o necessario) suddividere gli agenti in classi di priorità?

Incrociando le risposte a queste domande con i risultati esposti nel capitolo precedente si dovrebbe essere in grado di eliminare alcuni algoritmi che non sembrano adatti alla soluzione

del DCOP. Rimane comunque la necessità di effettuare ulteriori prove per decidere quale algoritmo è migliore per la rete in questione.

È più difficile stabilire se sia necessario creare un nuovo algoritmo per risolvere il tipo di DCOP di cui ci si è occupati in questo lavoro. È indubbio che il DCOP in questione sia molto particolare: la presenza contemporanea di vincoli *hard* e *soft* e di vincoli binari ed n-ari caratterizzano il problema. Se, dopo ulteriori esperimenti, si giungesse alla conclusione che nessun algoritmo esistente è in grado di risolvere il problema con prestazioni giudicate sufficienti allora si consiglierebbe la creazione di un nuovo algoritmo specifico per questo tipo di DCOP.

6.2 Lavori futuri

Tutti gli esperimenti presentati sono stati eseguiti utilizzando la modalità "centralizzata" di FRODO; ciò significa che tutti gli agenti vengono eseguiti all'interno della stessa *JVM* (Java Virtual Machine) e che comunicano attraverso uno scambio di puntatori. Questo non inficia i risultati dei test perché gli algoritmi eseguiti sono sempre algoritmi distribuiti ma i risultati non tengono conto di eventuali problemi caratteristici dei sistemi distribuiti.
In una rete reale, ad esempio, agenti diversi verrebbero eseguiti su macchine diverse e comunicherebbero tramite rete; essendo questa situazione molto diversa da quella sopra esposta sarebbe utile effettuare degli esperimenti che mostrano il comportamento del modello in un caso veramente distribuito. Esperimenti così condotti permetterebbero di valutare gli algoritmi in situazioni più simili alla vita reale e consentirebbero una migliore comprensione di come inconvenienti quali congestione di rete e perdita di pacchetti influenzano la soluzione del problema.
Un altro problema da risolvere è la valutazione della stabilità di una soluzione: in una rete molto dinamica potrebbe essere necessario cambiare spesso i valori delle variabili in modo da "inseguire" la soluzione ottima a fronte di cambiamenti dei parametri del problema, come ad esempio la potenza disponibile. Molti componenti elettrici, però, subiscono effetti negativi se sottoposti a continui transitori; in alcuni casi si potrebbe addirittura arrivare al paradosso di un carico che è continuamente in transitorio perché la soluzione cambia troppo rapidamente.
È perciò auspicabile riuscire a trovare un modo per preservare la stabilità di una soluzione, eventualmente anche a scapito della soluzione ottima. Un modo per ottenere questo risultato

sarebbe quello di imporre ad ogni agente un vincolo unario *soft* il cui costo dipende dalla differenza tra la soluzione attuale e quella precedente.

Tale soluzione genera però due problemi, uno di carattere pratico e uno di carattere teorico. Il problema di carattere pratico consiste nel fatto che l'aggiunta di vincoli potrebbe complicare il problema e renderlo più difficile da risolvere. Il problema di carattere teorico è dovuto all'interazione tra questi nuovi vincoli e quelli preesistenti; i nuovi vincoli, infatti, potrebbero causare un allontanamento eccessivo dalla soluzione ottimale per privilegiare la stabilità della soluzione. Una possibile soluzione consiste nel pesare i costi generati dai nuovi vincoli per un numero inferiore a 1 in modo da diminuirne l'importanza. Ciò richiede, però, nuovi esperimenti al fine di individuare il valore ottimale da usare per pesare i costi; è lecito inoltre supporre che tale parametro dipenderebbe dal tipo di rete gestita e dai carichi che la popolano, il che implica l'introduzione di una nuova variabile nel modello.

Un'altra idea da sviluppare consiste nel partizionare la rete in modo da rendere più semplice la soluzione del problema; come illustrato nel Capitolo 5 un numero elevato di agenti complica la soluzione del problema fino a renderlo impossibile. Una possibile soluzione al problema consiste nel partizionare la rete in sotto-reti e considerare ogni sotto-rete come un problema a sé stante.

Ciò significa che la potenza disponibile viene fornita a “macro-agenti” che non rappresentano dei veri carichi ma dei sotto-sistemi della rete; all'interno di ogni sotto-rete la potenza viene distribuita ai carichi in base al modello valutato in questa tesi. Ovviamente è possibile applicare questa idea ricorsivamente in modo da creare gerarchie più complicate.

Sarebbe quindi interessante capire se è possibile modificare il modello in modo da implementare questa idea e quali difficoltà sorgerebbero. Ad esempio si renderebbe necessario valutare che impatto avrebbero le classi di priorità in questo modello, oppure come i tempi necessari per giungere alla soluzione si dilatano man mano che la gerarchia sopra illustrata diventa più complicata.

Bibliografia

[1] Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, e Nick Jennings. Agent-Based Control for Decentralised Demand Side Management in the Smart Grid. In, *Proceedings of the the Tenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011), Taipei, Taiwan, 02 - 06 Maggio 2011*, 5-12.

[2] Yoseba K. Penya e Nicholas R. Jennings. Optimal Combinatorial Electricity Markets. *International Journal of Web Intelligence & Agent Systems, 6, (2), 123-135. (2008)*

[3] Sofia Ceppi e Nicola Gatti. An algorithmic game theory study of wholesale electricity markets based on central auction. *Integr. Comput.-Aided Eng., 17:273-290, Dicembre 2010.*

[4] Miroslav Prýmek e Aleš Horák. Multi-agent approach to power distribution network modelling. *Integr. Comput.-Aided Eng., 17:291-303, Dicembre 2010.*

[5] Maria J. Santofimia, Xavier del Toro, Pedro Roncero-Sánchez, Francisco Moya, Miguel A. Martinez e Juan C. Lopez. A qualitative agent-based approach to power quality monitoring and diagnosis. *Integr. Comput.-Aided Eng., 17:305-319, Dicembre 2010.*

[6] Sara Mullen e Getiria Onsongo. Decentralized agent-based underfrequency load shedding. *Integr. Comput.-Aided Eng., 17:321-329, Dicembre 2010.*

[7] Jignesh M. Solanki, Sarika Khushalani Solanki e Noel Schulz. Multi-agent-based reconfiguration for restoration of distribution systems with distributed generators. *Integr. Comput.-Aided Eng., 17:331-346, Dicembre 2010.*

[8] Michele Tartara. Cooperative Multiagent Systems for Electrical Energy Management. *Technical Report 2011.10, Dipartimento di Elettronica e Informazione, Politecnico di Milano, Marzo 2011.*

[9] Makoto Yokoo, Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-Agent Systems. *Springer, 2001.*

[10] Manuale di FRODO, http://thomas.leaute.name/frodo/manual/FRODO_User_Manual.html. *URL consultato il 27 Marzo 2012.*

[11] Organising Committee of the Third International Competition of CSP Solvers. XML

Representation of Constraint Networks – Format XCSP 2.1, 15 *Gennaio 2008.* http://www.cril.univ-artois.fr/~lecoutre/research/benchmarks/benchmarks.html

[12] Pragnesh J. Modi, W Shen, Milind Tambe e Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence, 161:149–180, 2005.*

[13] Adrian Petcu e Boi Faltings. DPOP: A Scalable Method for Multiagent Constraint Optimization. In, *Leslie Pack Kaelbling and Alessandro Saffiotti, editors, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), pages 266–271, Edinburgh, Scotland, 31 Luglio – 5 Agosto 2005. Professional Book Center, Denver, USA.*

[14] Brammert Ottens e Boi Faltings. Coordinating Agent Plans Through Distributed Constraint Optimization. In, *Proceedings of the ICAPS'08 Multiagent Planning Workshop (MASPLAN'08), Sydney, Australia, 14 Settembre 2008.*

[15] Adrian Petcu e Boi Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In, *Manuela M. Veloso, editor, Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07), pages 1452–1457, Hyderabad, India, 6–12 Gennaio 2007.*

[16] Adrian Petcu e Boi Faltings. O-DPOP: An algorithm for open/distributed constraint optimization. In, *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI'06), pages 703–708, Boston, Massachusetts, U.S.A., 16–20 Luglio 2006. AAAI Press.*

[17] Katsutoshi Hirayama e Makoto Yokoo. Distributed partial constraint satisfaction problem. In, *Gert Smolka, editor, Proceedings of the Third International Conference on Principles and Practice of Constraint Programming (CP'97), volume 1330, pages 222–236, Linz, Austria, 29 Ottobre – 1 Novembre 1997.*

[18] Weixiong Zhang, Guandong Wang, Zhao Xing e Lars Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Journal of Artificial Intelligence Research (JAIR), 161(1–2):55–87, Gennaio 2005.*

[19] Rajiv T. Maheswaran, Jonathan P. Pearce e Milind Tambe. Distributed algorithms for DCOP: A graphical-game-based approach. In, *David A. Bader and Ashfaq A. Khokhar,*

editors, Proceedings of the ISCA Seventeenth International Conference on Parallel and Distributed Computing Systems (ISCA PDCS'04), pages 432–439, Francisco, California, USA, 15–17 Settembre 2004. ISCA.

[20] Alessandro Farinelli, Alex Rogers, Adrian Petcu e Nicholas R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In, *Lin Padgham, David C. Parkes, Jörg P. Müller, and Simon Parsons, editors, Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08), pages 639–646, Estoril, Portugal, 12–16 Maggio 2008.*

[21] Marius-Călin Silaghi. Hiding absence of solution for a distributed constraint satisfaction problem (poster). In, *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS'05), pages 854–855, Clearwater Beach, FL, USA, 15–17 Maggio 2005. AAAI Press.*

[22] Marius-Călin Silaghi e Debasis Mitra. Distributed constraint satisfaction and optimization with privacy enforcement. In, *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'04), pages 531–535, Beijing, China, 20–24 Settembre 2004. IEEE Computer Society Press.*

[23] Boi Faltings, Thomas Léauté e Adrian Petcu. Privacy Guarantees through Distributed Constraint Satisfaction. In, *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'08), pages 350–358, Sydney, Australia, 9–12 Dicembre 2008.*

[24] Thomas Léauté e Boi Faltings. Privacy-Preserving Multi-agent Constraint Satisfaction. In, *Proceedings of the 2009 IEEE International Conference on PrivAcy, Security, riSk And Trust (PASSAT'09), pages 17–25, Vancouver, British Columbia, 29–31 Agosto 2009. IEEE Computer Society Press.*

[25] Thomas Léauté e Boi Faltings. Coordinating Logistics Operations with Privacy Guarantees. In, *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI'11), pages 2482–2487, Barcelona, Spain, 16–22 Luglio 2011. AAAI Press.*